

# TRANSIENT STATE ANALYSIS AND OPTIMIZATION ON POWER PLANT START-UP ROUTINE USING PSO

LEE YI YANG



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**TRANSIENT STATE ANALYSIS AND OPTIMIZATION ON  
POWER PLANT STARTUP ROUTINE USING PSO**

**LEE YI YANG**

**This report is submitted in partial fulfilment of the requirements  
for the degree of Bachelor of Computer Engineering with Honours**

**Faculty of Electronic and Computer Technology and Engineering  
Universiti Teknikal Malaysia Melaka**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**2024**

**BORANG PENGESAHAN STATUS LAPORAN  
PROJEK SARJANA MUDA II**

Tajuk Projek : Transient State Analysis and Optimization on Power Plant Startup Routine Using PSO  
Sesi Pengajian : 2023/2024

Saya LEE YI YANG mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

**SULIT\***

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

**TERHAD\***

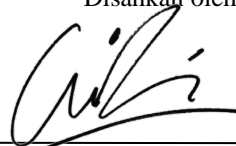
(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.)

**TIDAK TERHAD**

Disahkan oleh:



(TANDATANGAN PENULIS)



(COP DAN TANDATANGAN PENYELIA)

Alamat Tetap: SP1438, Jalan  
Bukit Emas 4,  
Taman Bukit  
Emas, 78000 Alor  
Gajah, Melaka

Tarikh : 24 January 2024

**PROF. MADYA DR. WIRA HIDAYAT BIN MOHD SAAD**  
Profesor Madya  
Fakulti Kejuruteraan Elektronik Dan Kejuruteraan Komputer  
(FKEK)  
Universiti Teknikal Malaysia Melaka (UTeM)  
Hang Tuah Jaya  
76100 Durian Tunggal, Melaka, Malaysia  
06-270 2394 / 012-916 1372  
wira\_yi@utem.edu.my


Tarikh : 24 January 2024

24 January 2024

## DECLARATION

I declare that this report entitled “Transient State Analysis and Optimization on Power Plant Startup Routine Using PSO” is the result of my own work except for quotes as cited in the references.



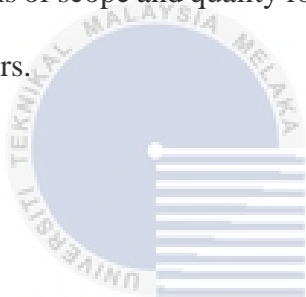
Signature :  .....

Author : Lee Yi Yang .....

Date : 24 January 2024 .....

## APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Computer Engineering with Honours.



Signature : 

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Supervisor Name : Professor Madya Dr. Wira Hidayat bin Mohd Saad

Date : 24 January 2024

## DEDICATION

To my beloved mother and father. To the experiences I would not trade for anything  
in this world.



## ABSTRACT

A conceptual novel method of optimizing power plant startup time is proposed in this project to minimize the startup cost, synchronization problem and human error associate with it. The problem that is long unoptimized power plant startup is approached with a two phased solution that is transient state analysis and PSO model. The purpose of the project is to explore new solution to the power plant startup problem other than using dynamic model from power plant simulator. The project can be further divided into four components namely PSO, Data Analysis, manual work and Display GUI. The PSO is tasked with producing a set of parameters that can be used to fine tune the power plant in order to produce predicted power level. Whereas the Data Analysis part serves to model the power plant's behaviour into a polynomial equation. All of the work involved in the project is done in MATLAB with the exception of manual work which is carried out in Microsoft Excel.

## ABSTRAK

*Kaedah baru konseptual untuk mengoptimumkan masa permulaan loji kuasa dicadangkan dalam projek ini untuk meminimumkan kos permulaan, masalah penyegerakan dan ralat manusia yang dikaitkan dengannya. Masalah yang merupakan permulaan loji janakuasa yang lama tidak dioptimumkan didekati dengan penyelesaian dua fasa iaitu analisis keadaan sementara dan model PSO. Tujuan projek ini adalah untuk meneroka penyelesaian baharu kepada masalah permulaan loji janakuasa selain daripada menggunakan model dinamik daripada simulator loji kuasa. Projek ini boleh dibahagikan lagi kepada empat komponen iaitu PSO, Analisis Data, kerja manual dan GUI Paparan. PSO ditugaskan untuk menghasilkan satu set parameter yang boleh digunakan untuk memperhalusi loji kuasa untuk menghasilkan tahap kuasa yang diramalkan. Manakala bahagian Analisis Data berfungsi untuk memodelkan tingkah laku loji kuasa ke dalam persamaan polinomial. Semua kerja yang terlibat dalam projek ini dilakukan dalam MATLAB dengan pengecualian kerja manual yang dijalankan dalam Microsoft Excel.*



## ACKNOWLEDGEMENTS

I would like to express my utmost gratefulness to my parents and family which has aid me in every way possible. Words cannot describe how much has gone into supporting me on my academic journey. My gratitude also goes to my supervisor of this project, Professor Madya Wira Hidayat, who has been my main enabler throughout the research. He was always ready with suggestions and pragmatic ideas regarding the project. The progress I have made is owe by his constant supervision in the form of fortnightly meetings. Besides, I would like to extend my thanks to all my friends who have always been with me through thick and thin. To Tey Eyson who always brings laughter and joy wherever you go. To Ng Chertat who constantly aid me in my studies. To all the friends I made along the way where their path intercepts mine, thank you. To everyone whom I share this experience with, thank you and I wish nothing but the best for you.

## TABLE OF CONTENTS

<b>Declaration</b>	
<b>Approval</b>	
<b>Dedication</b>	
<b>Abstract</b>	i
<b>Abstrak</b>	ii
<b>Acknowledgements</b>	iii
<b>Table of Contents</b>	iv
<b>List of Figures</b>	ix
<b>List of Tables</b>	xi
<b>List of Symbols and Abbreviations</b>	xii
<b>List of Appendices</b>	xiii
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Project Overview	2
1.2 Problem Statement	2
1.3 Objectives and Scope	4
1.3.1 Objectives	4

1.3.2	Scope of Work	5
<b>CHAPTER 2 BACKGROUND STUDY</b>		<b>7</b>
2.1	Power Generator	8
2.2	CCGT Power Plant	9
2.3	Types of Start Up	11
2.3.1	Hot Start	11
2.3.2	Warm Start	12
2.3.3	Cold Start	12
2.4	Basic Start Up Routine	13
2.4.1	Gas Turbine Initiate Start	14
2.4.2	Purging	14
2.4.3	Firing	15
2.4.4	Warm Up	15
2.4.5	Full Speed No Load (FSNL)	16
2.4.6	Grid Synchronization	16
2.4.7	Temperature Matching	17
2.4.8	Steam Turbine Coupling	17
2.4.9	Inlet Pressure Control	18
2.4.10	Back Pressure Enable	18
2.4.11	Temperature Matching Off	18

2.4.12 Sequence Complete	18
2.4.13 Minimum Load	18
2.4.14 Baseload/ Declare Available	19
2.5 Particle Swarm Optimization	19
2.6 Optimizer Model and Working Principle	21
2.7 Improvement on Existing System	21
2.8 Optimization System in Various Fields	23
2.9 Related Studies	24
<b>CHAPTER 3 METHODOLOGY</b>	<b>26</b>
3.1 Research Methodology	27
3.2 Data Collection	28
3.3 Data Pre-processing	31
3.4 Data Analysis	34
3.5 MATLAB	35
3.6 Optimization of Start-Up Routine	36
3.6.1 Setting of Parameters	36
3.6.2 Reducing the Process Time	38
<b>CHAPTER 4 RESULTS AND DISCUSSION</b>	<b>40</b>
4.1 Result Summary	41
4.2 Manual Work	42

4.3	Coding Analysis	43
4.3.1	Particle Swarm Optimization Model	43
4.3.2	Data Analysis	45
4.3.3	Graphical User Interface (GUI)	46
4.4	Sensor Integration	47
4.5	Output Interpretation	49
4.6	Comparison Between Optimized and Unoptimized Start-Up Time	51
4.7	Time Saved	53
4.8	Impact of PSO Parameters on Optimized Curve	54
4.9	Accuracy of the Particle Swarm Model	57
4.9.1	Residual Sum of Squares (RSS)	57
4.9.2	Mean Error	59
4.10	Discussions	60
4.11	Research Limitation	61
4.12	Research Sustainability	62
	<b>CHAPTER 5 CONCLUSION AND FUTURE WORKS</b>	<b>64</b>
5.1	Conclusion	65
5.2	Future Works	65
	<b>REFERENCES</b>	<b>67</b>
	<b>APPENDICES</b>	<b>73</b>



## LIST OF FIGURES

<i>Figure 2.1 CCGT Power Generator with Heat Recovery Steam Generator</i>	10
<i>Figure 2.2 Graph of Power output against Time Stamp of Hot Start</i>	11
<i>Figure 2.3 Graph of Power output against Time Stamp of Warm Start</i>	12
<i>Figure 2.4 Graph of Power output against Time Stamp of Cold Start</i>	13
<i>Figure 2.5 Basic Startup Sequence</i>	14
<i>Figure 2.6 Graphical Display of particle converging on a possible solution</i>	20
<i>Figure 2.7 Screening Curves of Total Cost against Operating hours per Year</i>	22
<i>Figure 3.1 Flowchart of Research Methodology</i>	27
<i>Figure 3.2 Sample Excerpt of Start-Up data provided by the company</i>	30
<i>Figure 3.3 Processes of Data Pre-Processing</i>	33
<i>Figure 4.1 Block Diagram</i>	41
<i>Figure 4.2 Design Plan</i>	42
<i>Figure 4.3 Updating with Value Closest to Target Value and One-Time Nested Condition Checking Loop</i>	44
<i>Figure 4.4 Sample Output of PSO Model</i>	45
<i>Figure 4.5 Curve Fitting</i>	46
<i>Figure 4.6 Code Snippet of GUI</i>	47
<i>Figure 4.7 Declaration of Arduino Objects</i>	48
<i>Figure 4.8 Sensor Value Mapping</i>	48

<i>Figure 4.9 Construction of Sensor Integration with Accelerometer and Arduino UNO</i>	49
<i>Figure 4.10 Output with Highlighted Explanation</i>	50
<i>Figure 4.11 Accuracy Metrics</i>	51
<i>Figure 4.12 num_particles = 100, w = 0.7, c1,c2 = 1.5</i>	55
<i>Figure 4.13 num_particles = 50, w = 0.7, c1,c2 = 1.5</i>	55
<i>Figure 4.14 num_particles = 10, w = 0.7, c1,c2 = 1.5</i>	56
<i>Figure 4.15 num_particles = 50, w = 0.5, c1,c2 = 1.5</i>	56
<i>Figure 4.16 num_particles = 50, w = 0.9, c1,c2 = 1.5</i>	56
<i>Figure 4.17 num_particles = 50, w = 1.0, c1,c2 = 1.5</i>	57
<i>Figure 4.18 Formula of RSS Calculation</i>	58





## LIST OF TABLES

Table 4.1: Output Curves for Initiate and Purging Sequence	51
Table 4.2: Timesaved for Each Sequence	53
Table 4.3: RSS of Each Sequence	59
Table 4.4: Mean Error of Each Sequence	59



## LIST OF SYMBOLS AND ABBREVIATIONS

PSO	:	Particle Swarm Optimization
DC	:	Direct Current
AC	:	Alternating Current
CCGT	:	Combined Cycle Gas Turbine
GT	:	Gas Turbine
FSNL	:	Full Speed No Load
TNH	:	Actual Turbine Speed
HP	:	High Pressure
HRSG	:	Heat Recovery Steam Generator
PI	:	Plant Information
CSV	:	Comma-Separated Values
GUI	:	Graphical User Interface
ANOVA	:	Analysis of Variance
UI	:	User Interface
PC	:	Personal Computer
RSS	:	Residual Sum of Squares
ARIMA	:	Autoregressive Integrated Moving Average

## LIST OF APPENDICES

Appendix A: PSO Sample Coding for Firing Sequence of Cold Startup	74
Appendix B: Data Analysis Sample Coding for Firing Sequence of Cold Startup	77
Appendix C: Display GUI Sample Coding	77



# CHAPTER 1

## INTRODUCTION



This chapter aims to outline the objective and aims of the project Transient State Analysis and Optimization Model on Power Plan Startup Routine. This chapter also highlights the problems that lead to the need to develop this project. Besides this chapter defines the outline of this project which includes data collection and analysis, model development and lastly application development.

## 1.1 Project Overview

This project aims to conduct a transient state analysis and optimization of the power plant startup routine. The aim of this project is to cut the time required for the power plant to reach steady state by analyzing power plant behavior in transient state and then apply PSO to obtain best parameter, by that it can minimize power consumption during startup, and reduce equipment wear and tear.

The first phase of the research will involve data collection and analysis based on previous startup routine, after that the behavior of each startup sequence will be modelled in the form of a mathematical model. Emphasis will also be given to suggest mitigative actions to power plant by PSO model in order to improve startup time. It may or may not include modifying startup sequence, adjusting power plant parameters, or implementing new technologies.

A display GUI which shows all the details regarding the PSO model and Data Analysis will also be developed to allow users to understand the principles of PSO. Accuracy test will be conducted to evaluate the goodness of the modelled function and performance test will be carried out as well.

The ultimate goal of the project is to improve the efficiency, reliability, and safety of the power plant, leading to cost savings and improved performance.

## 1.2 Problem Statement

Startup of a power plant is an irreplaceable process that is vulnerable to a plethora of problems that can bring substantial impact on the efficiency, safety and cost associated with the plant. The unoptimized startup routine is not ideal due to its time-

consuming, energy-intensive. Moreover, due to prolonged start time the stress point existing in the power plant will be unstable for a period longer than usual. One such example is the thermal stress points in pressure components which in transient state will have multiple stress point which relocate throughout the process. Unoptimized startup leads to increased operating costs and compromised equipment lifespan. Although the cost associated with start-up wear and tear is often subjective to the model of power generator[1], Additionally, the synchronization of the power signal of the generator with the on-grid signal is a major challenge with consideration such as the phase sequence, voltage magnitude, frequency, and phase angle[2].

The power plant startup routine is a complex [3] and intricate process that requires careful planning, execution, and monitoring. However, despite its importance, this process is often prone to several problems that can have a significant impact on the efficiency, safety, and cost-effectiveness of the power plant. One of the main challenges in the startup routine is the time-consuming and energy-intensive nature of the process, which can lead to significant costs in terms of fuel consumption and labor. Additionally, the startup routine can cause significant wear and tear on equipment, leading to increased operating costs and reduced equipment lifespan.

Furthermore, the current startup routine often involves manual procedures that can be prone to errors, such as misalignments, miscommunications, or misconfigurations, which can result in equipment damage, accidents, or even power outages [4] . These problems can be especially problematic in power plants that operate in remote or hazardous environments, where the risk of accidents and equipment failure is particularly high. That is why a control system needs to be in place to minimize those problems.

Power plant operators and engineers are currently exploring novel and innovative method to improve startup with the combination of automation, monitoring and controller. By implementing these technologies, it is possible to reduce the time, energy, and labor required for the startup process[5], while also minimizing the risk of errors and equipment damage. The efficiency, safety, and cost-effectiveness of power plant operations can be increased with those efforts, ensuring stable and sustainable power generation for years to come.

Besides, there are also factors such as synchronizing the generated power to the on-grid signal which poses some difficulties. Few of the obstacles in this process comes in the form of phase sequence, voltage magnitude, frequency, and phase angle. It is crucial that the generator's signal matched the on-grid signal. Serious consequences, such as electrical fires, equipment damage, or even widespread power outages [4] can arise as a result of synchronization failure so it is important to steer clear of the issue. Despite these challenges, however, the successful synchronization of generator power signals with on-grid signals is essential to maintaining a reliable and efficient power supply, particularly in areas with high demand for electricity.

### **1.3 Objectives and Scope**

#### **1.3.1 Objectives**

The aim of this project is to develop a control system model to analyze the transient state and optimize the startup routine of Power Plan. To achieve that, following objectives need to be accomplished:

- a) To collect and analyze the data from previous Power Plan Startup Routine.
- b) To achieve synchronization between the power signal generated by the generator and the on-grid signal.

- c) To propose a procedure of mitigative action to improve the startup time.

### 1.3.2 Scope of Work

First phase of the project is to perform data collection and analysis. The data will be provided to researcher as this project is a collaboration with Malakoff Corporation Berhad. Revision of past data will be carried out in the form of MATLAB coding to identify any cause of long startup and areas for improvement. This phase is also identified as data pre-processing.

After data pre-processing, construction of a mathematical model will begin by leveraging curve fitting to replicate the behavior of the plant at each sequence. The formula which is the output of the model will take in various parameters which is measured variables, controllable variables and produce an output which in this case the power level. A set of parameters at each time interval will also be the final product of this phase.

The control system will be revised to synchronize the power signal of the generator with the on-grid signal, reducing the risk of equipment damage, safety hazards, and power outages. Various optimization strategies will be considered to improve startup time and reduce power consumption. This may or may not include adjusting equipment settings, changing operational procedures, or implementing new technologies.

The effectiveness of the PSO model will be evaluated with performance metric as well as the accuracy metrics. The GUI built will also enable the authorized personnel from power plants to understand the impact of the project on the startup time.



The application which is the Display GUI will display various status of the sequence including the mathematical model, dataset and the improvement from unoptimized startup. The streamlined and direct presentation of power plant stat will allow human operators to work more efficiently.

Overall, the scope of work which encompasses four component Manual Work, PSO model, Display GUI and Data Analysis is focused on optimizing the power plant with efficiency, safety, and cost-effectiveness as the main goals of the optimization.



## CHAPTER 2

### BACKGROUND STUDY



The chapter serves to conceptualize the basic principles that are important throughout the development process of the research project. It includes basic working principles of power plant, Start-up type and sequences. Similar works that have been studied also will be covered in this chapter to provide a means of comparison to this project. Emphasis will also be given to existing technology used in the optimization process. A review of previous literature will also be carried out.

## 2.1 Power Generator

The generation and distribution of electricity are an important aspect of modern human civilization, it provides a mean of energy transfer for electric equipment utilization. Power generator which is at the starting point of the generation plays an irreplaceable role to convert mechanical energy from chemical energy from the fuel to electrical energy. It is commonly used in a variety of settings such as power plants, industrial facilities, and emergency backup systems [6]. The operating principle behind a power generator is electromagnetic induction of Faraday's law, it dictates that when an electrical conductor is passing through a magnetic field, an electric signal will be induced in the conductor. Power generators can be categorized into two main types: AC generators and DC generators[7]. The most common generator which is AC generator generate AC current which is easy to distribute and require an adapter to convert it back into DC current. DC generators, on the other hand, are designed to produce direct current, which is used in applications such as battery charging and electroplating.

One of the most common types of power generators used today is the gas turbine generator. This type of generator is based on the Brayton cycle[8], which is a thermodynamic cycle that uses air as the working fluid. Gas turbine generators can be used in both simple cycle and combined cycle configurations. In a simple cycle configuration, the byproduct such as the steam from outlet, which also have some leftover energy in it is not utilized resulting in inefficiency. In a combined cycle configuration, the gas turbine is combined with a steam turbine to produce additional power[9]. The more common variant of the two is combined cycle configuration because of the higher efficiency.

Another important aspect of power generators is the start-up process. Depending on the type of generator and the application, start-up processes can vary significantly[10]. Some generators are designed for continuous operation, while others are only operated intermittently. A sequential procedure is usually involved in the startup routine such as initialization, purging, firing, warming up, synchronization, and load increase. The entirety of the process can possibly take up to several hours, nevertheless it requires meticulous monitoring and control, any deviation in parameters will affect the outcome drastically.

A growing focus on sustainable and eco-friendly power generation solutions can be observed in recent years due to increased environmental awareness. This has driven researcher to invest more effort into developing renewable energy, energy storage, and optimization of various aspects of power plant. The crucial role of power generator has in meeting the world's ever-growing energy needs and technological advancements will continue to endure the test of time at least in the foreseeable future.

## 2.2 CCGT Power Plant

The low emissions and high efficiency of Combined Cycle Gas Turbine (CCGT) is the main advantages of the configuration of the power generator and has cemented the type of power plant as a reliable and efficient choice for energy generation. The type of power plant differs from normal power plant configuration as it consists of a gas turbine as well as a heat recovery steam generator (HRSG). The fuel which is usually natural gas is used to move the gas turbine, and the byproduct steam is recovered to move a secondary steam turbine. This way there are two point of energy generation in

the whole plant. The combination of these two cycles (gas and steam) results in a high efficiency that can reach up to 60% [11].

Flexibility is key in CCGT power plants to enable quick response to changes in grid demand. They have the ability to start and stopped relatively quicker than their counterparts, thus making them suitable for both base load and peaking operation. With this advantage it also comes with the disadvantage of having a higher number of startup and shut down. In addition, CCGT power plants emit significantly lower levels of pollutants such as NO<sub>x</sub>, SO<sub>x</sub>, and CO<sub>2</sub> than conventional fossil fuel power plants [12], making them relatively more environmentally friendly. A illustration of the CCGT is included in Figure 2.1 below.

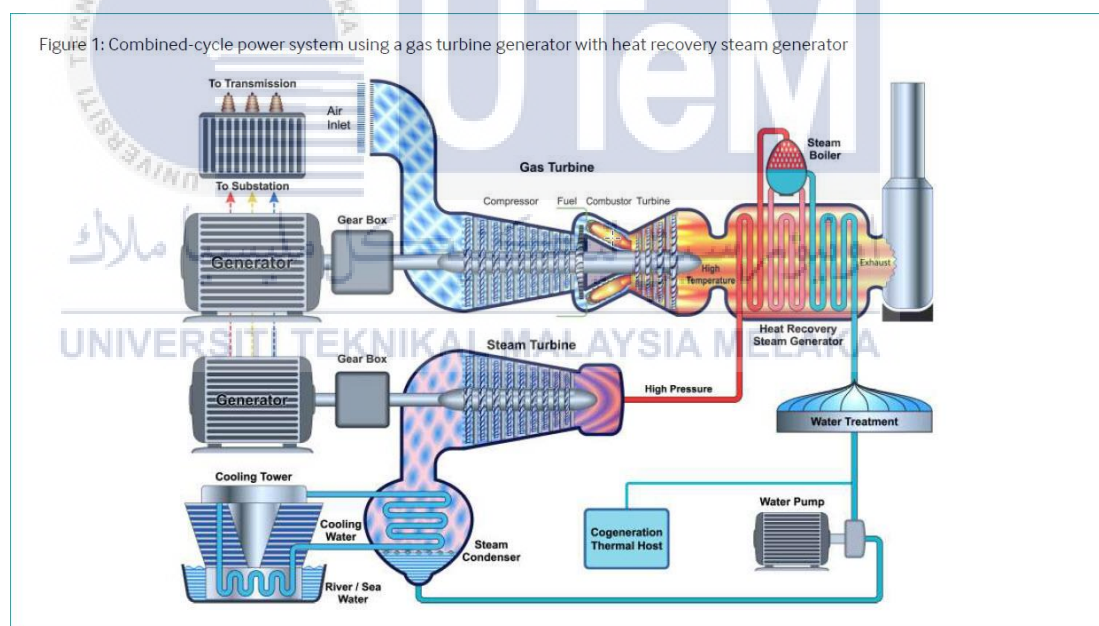


Figure 2.1 CCGT Power Generator with Heat Recovery Steam Generator

The use of CCGT power plants has become increasingly popular in the electricity industry due to their high efficiency, low emissions, and operational flexibility. They

are ideal for meeting the ever-increasing demand for electricity in a sustainable and eco-friendly manner. The growing emphasis on clean energy and reduction of greenhouse gas emissions is believed to drive adoption of more CCGT power plants in the near future.

## 2.3 Types of Start Up

### 2.3.1 Hot Start

Hot start is defined as a method of starting up a turbine where it is already at the operating temperature. The method is commonly used in restarting the power plant, in this case the gas turbine is shutdown, but the combustion chamber remains hot. This method is also efficient because the turbine is already at operating temperature, which means that there is no need for additional fuel to bring the turbine up to temperature[13]. Though it poses risks of damaging the turbine or other components of the power plant because the combustion chamber is not cooled down properly as it causes thermal and mechanical stress across the power plant. Figure 2.2 shows the power output plotted against time of hot start.

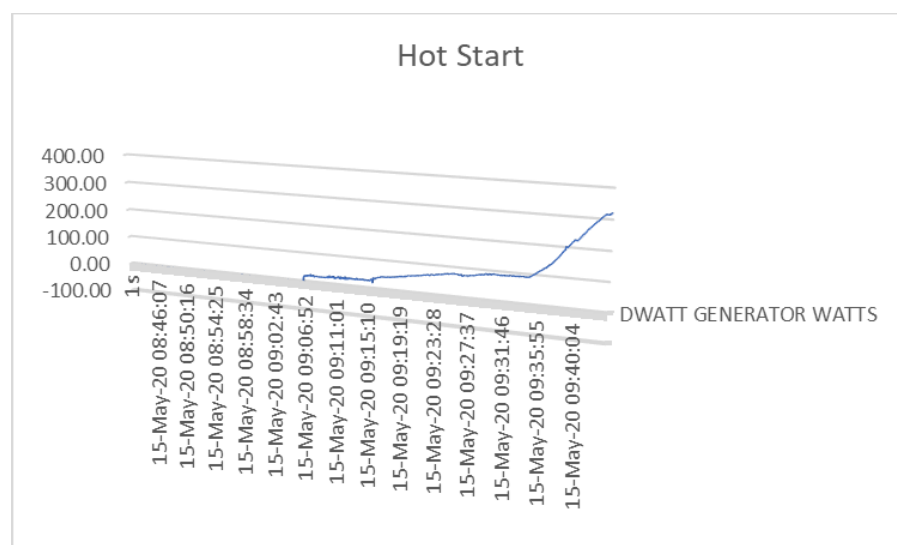


Figure 2.2 Graph of Power output against Time Stamp of Hot Start

### 2.3.2 Warm Start

A warm start is a type of start-up procedure for a power plant where the turbine and associated equipment are still at a high temperature, but the gas turbine has cooled down to some extent. In this start-up mode, the unit is brought online faster than in a cold start, but slower than in a hot start. This type of start-up procedure is often used when the power plant has been shut down for a relatively short period, such as overnight, and the equipment is still warm. Warm starts can help reduce wear and tear on the equipment and allow for faster ramp-up times. Figure 2.3 shows the power output plotted against time of warm start.

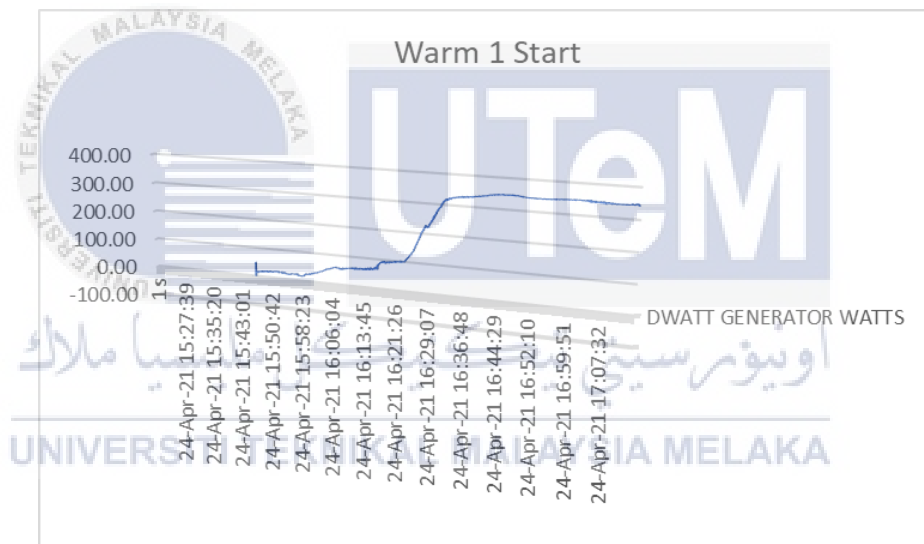


Figure 2.3 Graph of Power output against Time Stamp of Warm Start

### 2.3.3 Cold Start

In a cold start, the gas turbine has been completely shut down, and the temperature of all of its components has dropped to the ambient temperature. In this situation, the gas turbine has to be restarted from the beginning, which means that it has to be turned on and warmed up. The process of warming up the gas turbine components to their

operating temperature can take several hours, depending on the size and design of the turbine. During this process, the gas turbine is brought up to speed gradually, and the temperature of the various components is increased slowly to avoid thermal shock and damage. Once the gas turbine is up to its operating temperature, it can be synchronized to the grid and begin generating electricity. A cold start is typically the most time-consuming and energy-intensive type of start-up for a gas turbine. Figure 2.4 shows the power output plotted against time of cold start.

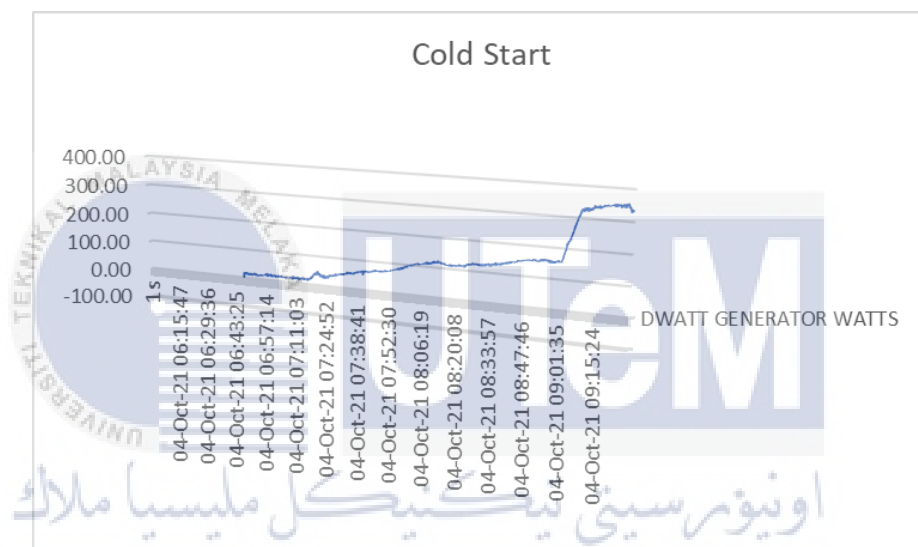


Figure 2.4 Graph of Power output against Time Stamp of Cold Start

## 2.4 Basic Start Up Routine

The start up routine is divided into fourteen sequences, it will be discussed in sequential order as in the actual power plant start up. The sequences have different runtime across each startup time and different behavior can also be observed too. A basic startup routine is shown in Figure 2.5.



## Start Up Sequence

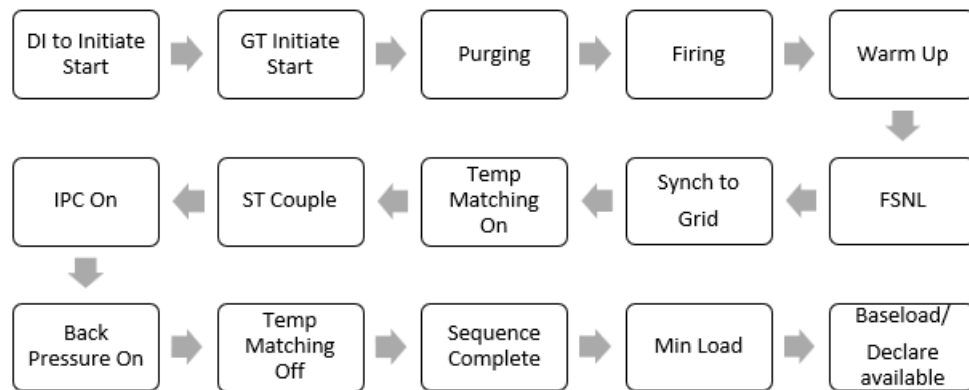


Figure 2.5 Basic Startup Sequence

### 2.4.1 Gas Turbine Initiate Start

In the first phase, the signal GT Initiate Start is sent to the gas turbine, operator is required to manually initiate start once all permissive are achieved. The permissive action includes the following, Steam Turbine ready to start, Aux. Steam pressure is greater than twelve bar, Aux Steam temperature is greater than 150 Celsius and etc.

### 2.4.2 Purging

Purging is a process used to clear the compressor of any residual gases before starting the combustion process. This is done to prevent the risk of a flashback, which can occur when the fuel ignites prematurely inside the combustion chamber, causing damage to the turbine blades and other components. Purging involves injecting air or an inert gas, such as nitrogen or carbon dioxide, into the combustion chamber to push out any residual gases. This process is typically automated and controlled by the plant's

control system to ensure that the purging is done safely and efficiently. The high-pressure turbine is at 23.45% speed (703rpm) for 11 minutes.

### 2.4.3 Firing

After purging, the vacant combustion chamber provides a place for fuel introduction. With the presence of fuel, the igniter or spark plug are activated to ignite the fuel. Throughout the combustion, the high-temperature gas escapes the combustion chamber and being directed to the gas turbine for power generation. The power plant is now generating power at a non-ideal state. High-pressure turbine is brought down to TNH=14% (420rpm).

### 2.4.4 Warm Up

Warm up sequence allows the high-pressure turbine to adapt to the speed and temperature for operation. It maintains the speed of the turbine at constant for approximately one minute. Firstly, by gradually increasing the operating temperature and reducing thermal stress it prolongs the lifespan of components. Secondly, it ensures optimal performance by allowing the generator to reach its ideal operating temperature[14], resulting in improved efficiency and fuel consumption. Thirdly, the sequence allow testing and calibrating of parameters from the control systems and safety mechanisms to take place thus increasing stability. Fourthly, it helps control emissions by allowing the generator to reach its optimal combustion temperature, minimizing environmental impact. Lastly, it extends the lifespan of the equipment and enhances safety by detecting any abnormalities or malfunctions before full operation.

Overall, the warm-up phase can be summarized as a buffer sequence for various fine-tuning of parameter to take place and is essential for efficient, dependable, and safe power generation.

#### **2.4.5 Full Speed No Load (FSNL)**

Full speed no load (FSNL) is a mode of operation for Combined Cycle Gas Turbine (CCGT) power plants, it starts when no electricity can be generated although the gas turbine is at its maximum speed. This is to stabilize the gas turbine and enable it to reach its maximum rotational speed. As with most of the few starting sequence in power plant startup it serves the purpose of increasing safety and efficiency during the startup process. Power generation is cut by disconnecting the gas turbine from the generator and the steam is bypassed around the heat recovery steam generator (HRSG) and discharged directly into the atmosphere without passing the steam turbine. The high-pressure turbine is at TNH 100% (3000rpm).

#### **2.4.6 Grid Synchronization**

In a combined cycle gas turbine (CCGT) power plant, after the gas turbine is fired and the speed has reached the rated speed, the generator is synchronized to the grid by using a generator circuit breaker (GCB). To ensure that the generator voltage and frequency match the grid synchronization is needed. To achieve stable synchronization, the sequence of the turbine and generator must be held, allowing time for the drum level to stabilize. The sequence may be hold for a longer period to ensure stability, in the optimization of the startup time, the faster the power plant reach this

sequence, the amount of time allowable for this sequence to take place is greater. After synchronization, the generator can be connected to the grid, and the load can be increased gradually[14].

#### **2.4.7 Temperature Matching**

It is important to maintain temperature matching between the gas turbine and steam turbine in order to maximize efficiency and power output. Temperature mismatches can lead to reduced efficiency and potential damage to the equipment. One way to achieve temperature matching is through careful control of the firing temperature in the gas turbine and the steam temperature in the heat recovery steam generator. Additionally, control systems can be implemented to monitor and adjust temperatures in real-time to ensure they remain within the desired range.

#### **2.4.8 Steam Turbine Coupling**

In this process, high pressure steam is admitted to the HP steam turbine by HRSG. This process requires a human operator to manually initiate the sequence in order to admit steam into the steam turbine. There are a few conditions that need to be met before this process can be commence. Some of a few conditions include High Pressure bypass valve should be 20 percent open and the High-Pressure header pressure should be no lesser than thirty-seven bar.

#### **2.4.9 Inlet Pressure Control**

Inlet pressure control is turned on and HP bypass is slowly close to increase pressure to the Steam Turbine.

#### **2.4.10 Back Pressure Enable**

IP and LP bypass is slowly close to increase pressure in the steam turbines. The back pressure will proceed in 30 seconds if the startup sequence is not set to HOLD.

#### **2.4.11 Temperature Matching Off**

In normal condition, if the start up sequence is not HOLD, the temperature matching will be switched off in 30 seconds after back pressure is enabled.

#### **2.4.12 Sequence Complete**

The sequence is completed after the temperature matching is switched off. The power generator is now in steady state.

#### **2.4.13 Minimum Load**

The minimum load of the generator varies based on the type of start-up and needs to be confirmed by a human operator. The operator is required to input the minimum load value.

#### 2.4.14 Baseload/ Declare Available

The load can now be increased to suit the needs of the client. The load might be slightly under the desired power level due to the ambient temperature.

### 2.5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the social behavior of birds in a flock or fish in a school[15]. The algorithm sustains a population of candidates of solution also known as particles. The parameters of the particles can be tweaked to suit different optimization problem. The particles will then drift around the predetermined search space bounded by an upper and lower boundary and search for the position with best cost value. The position and velocity will be adjusted based on previous best personal and global position every iteration.

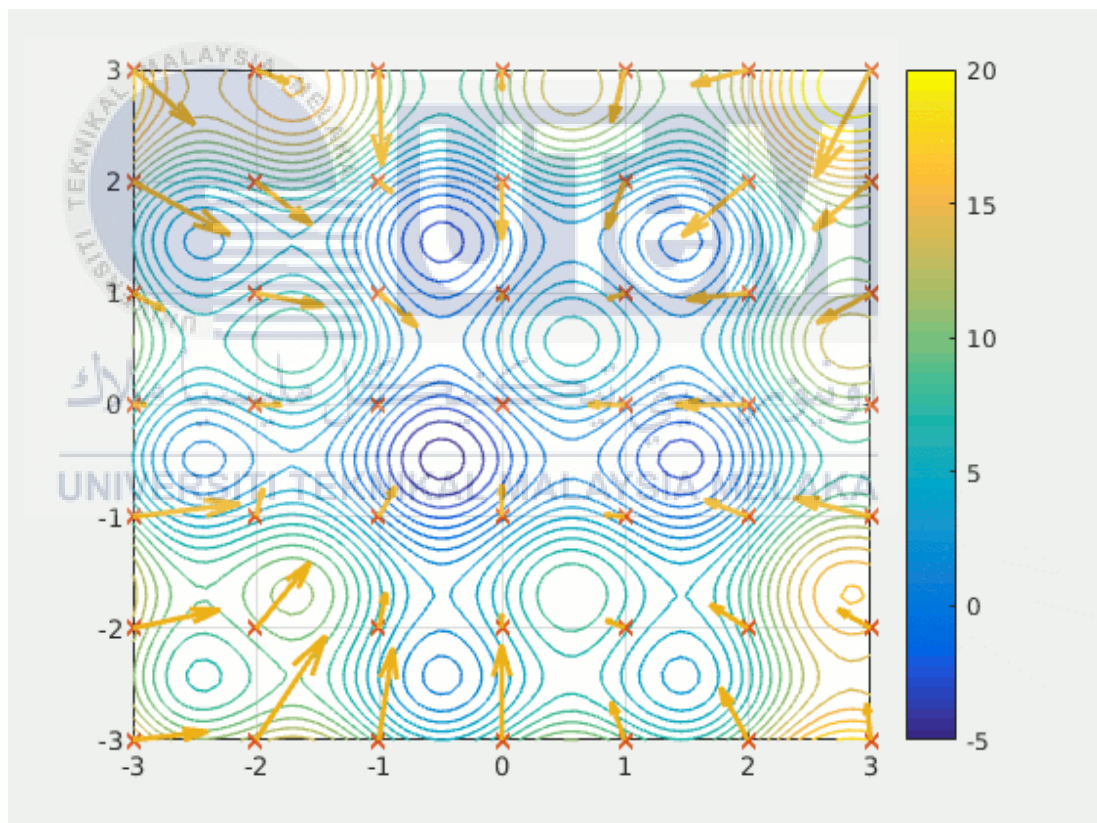
Each particles position represents a possible solution for the problem, and its fitness is calculated with a formula known as objective function. The objective function determines how good or bad the particle's position is, and the goal is to find the particle that has the best fitness value, i.e., the global optimum. The global optimum can either be a global maximum or global minimum based on the problem one is trying to solve.

First, the algorithm initializes a population with size specified by user, and assign random position and velocity bound by the search space. Then, it updates the velocity and position of each particle based on its own best-known position and the best-known position of the swarm[15]. The velocity update equation consists of two terms: the cognitive component, which is the particle's memory of its own best-known position, and the social component, which is the swarm's memory of the best-known position

so far. These terms are weighted by two constants called acceleration coefficients those constants dictate how much the particles are affected by their previous best position and the global best position.

At each iteration the velocity and position value are replaced if better fitness value is achieved or if the same or worse fitness value is obtained, the position remains the same as the previous iteration. The iteration is stopped when the stopping criteria is met. Finally, the particle with the best fitness value is returned as the optimal solution.

Figure 2.6 display the convergence of particle with animation for visualization.



*Figure 2.6 Graphical Display of particle converging on a possible solution*

PSO has been successfully applied to various optimization problems in engineering, science, economics, and other fields. Its advantages include simplicity,

ease of implementation, and fast convergence to the global optimum. However, it is also prone to getting stuck in local optima and may require careful parameter tuning.

## 2.6 Optimizer Model and Working Principle

An optimizer model is a mathematical model that is designed to solve optimization problems. It is essentially a set of algorithms that aim to find the best solution to a particular problem by minimizing or maximizing an objective function[15]. The optimizer model works by iteratively adjusting the input parameters of the objective function until it finds the best combination of input parameters that result in the optimal output value of the objective function. In this way, it is able to search through a large number of possible solutions and converge towards the best possible solution. The optimizer model can be used in a wide range of applications, from engineering and finance to healthcare and logistics, to name just a few examples.



## 2.7 Improvement on Existing System

The main problem that the optimization algorithm intends to solve is the synchronization problem. As the current system does not have an optimizing algorithm in place, this results in a prolonged start up period because the control system needs to hold the power generator before it can close the circuit to sync with the power grid[16]. There are a few factors that act upon the time of power generator to achieve the desired signal namely the electromagnetics created by excitation circuit, the speed of the rotor and rotor spinning direction. Most of the factors stated above are dictated by the prime mover of the turbine which is combustion gas and steam. Besides, the



start-up period is largely restricted by the heat recovery steam generator (HRSG) and steam turbine generator (STG). The improvement on the start-up time can be achieved by using an algorithm that has the capability to process big data as the plant has been continuously generating lots of process and operating data including transient data associated with the CCGT unit start-up. All of these data are stored into PI historians and are analyzed routinely by O&M personnel for performance and reliability purposes. The significance of the optimizing algorithm is improving the start-up period that will in turn reduce the capital cost[4] and assist the plant to meet the Energy Commission (EC) requirement. Comparison of cost between OCGT, Coal, CCGT and Nuclear power plant is represented in Figure 2.7.

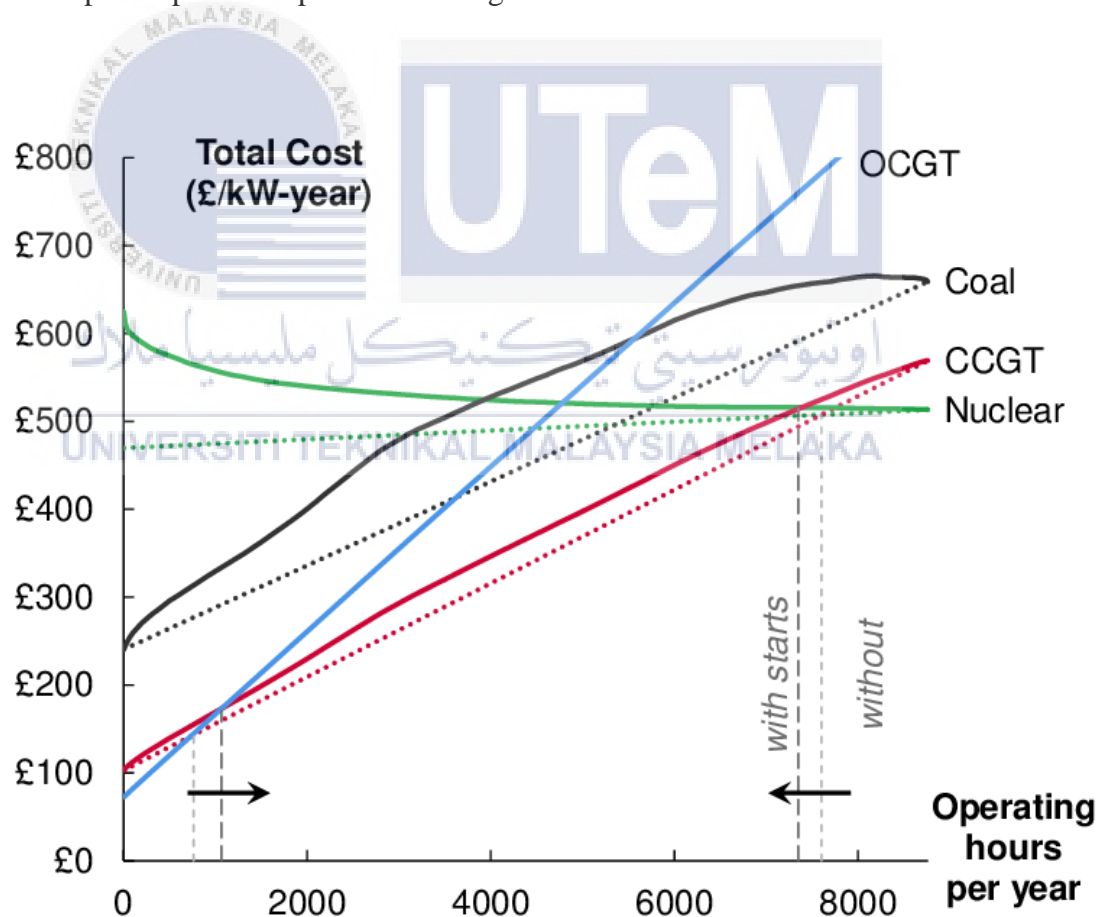


Figure 2.7 Screening Curves of Total Cost against Operating hours per Year

## 2.8 Optimization System in Various Fields

Optimization algorithms are used in a variety of fields to find the best solution to a given problem. In finance, optimization algorithms are used to determine the best investment strategy given certain constraints. For example, an investor may use optimization algorithms to determine the optimal mix of stocks, bonds, and other investments to maximize returns while minimizing risk.

In logistics, optimization algorithms are used to determine the most efficient way to transport goods from one location to another. For example, a delivery company may use optimization algorithms to determine the best route for its trucks to take to minimize travel time and fuel consumption.

In engineering, optimization algorithms are used to design and optimize various systems and processes[15]. For example, in mechanical engineering, optimization algorithms are used to design and optimize structures such as bridges and buildings. In chemical engineering, optimization algorithms are used to design and optimize chemical processes such as the production of drugs and other chemicals.

In machine learning, optimization algorithms are used to train models to make accurate predictions. For example, optimization algorithms such as gradient descent are used to minimize the error between the predicted outputs of a model and the actual outputs.

In summary, optimization algorithms have broad applications in various fields, and they can be used to solve a wide range of problems.

## 2.9 Related Studies

A study by Alessandro Nannarone and Sikke A. Klein titled “Start-Up Optimization of a CCGT Power Station Using Model-based Gas Turbine Control” focuses on a novel start-up procedure of existing combined cycle power station by applying a feedback loop to ensure that the stresses in steam turbine is reduced. The optimization is done in dynamic model that consists of blocks such as heat exchangers, valves, meters, sensors, and turbines. The model is also generally applicable to other power plant installations[17]. The feedback loop also controls the input such as steam turbine housing temperature. The startup time reduction of this study is 32.5% and 31.8% for cold and warm startup and the cost reduction is 47% and 32.4%[18].

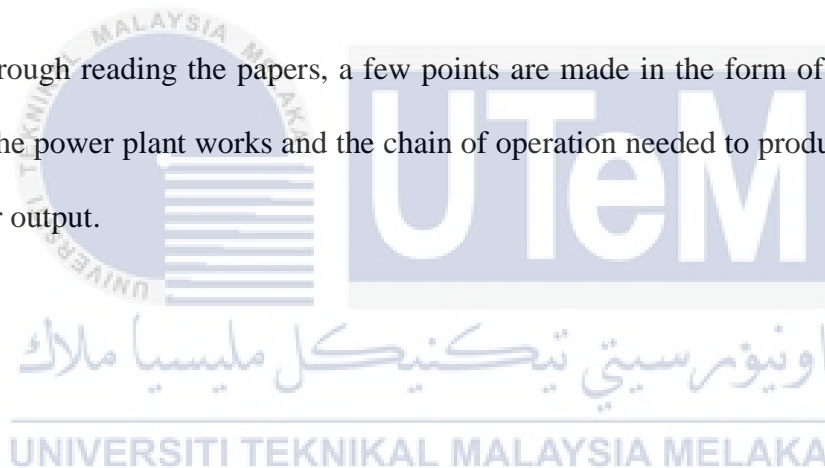
This study is different from the one in this thesis because it utilizes a dynamic model whereas in this thesis the main subject is data-driven optimization. The key takeaway from this study is that the input such as the fuel supply influence the power level substantially, and by increasing the input the ideal output power can be achieved.

Another study named “Particle Swarm Optimization: A Comprehensive Survey” by T. M. Shami highlights the impact of PSO parameters on the optimization result of the algorithm. It reviews the basic concepts of PSO, its variants, application and drawbacks of PSO and etc. The paper also shed light on the possibilities to hybridize the algorithm with other algorithms such as genetic algorithm (GA) and differential evolution (DE). It also states that PSO is simple to implement and code. There are three controlling parameters namely inertia weight, cognitive ratio and social ratio important in PSO[19].

This study has helped in understanding the basics of PSO and provide insight into how the algorithm should be applied to the particular problem that is power plant startup routine optimization for this thesis.

The paper by Hubel M and Meinke S titled “ Modelling and simulation of a coal-fire power plant for start-up optimization” also puts emphasis on the optimization with dynamic model building as the main method. The model included in the study can identify restrictions for quicker start-ups, with additional benefits of lesser fuel consumption and decreased emission all while having an acceptable amount of thermal and mechanical stress[20].

Through reading the papers, a few points are made in the form of understanding how the power plant works and the chain of operation needed to produce the optimal power output.



## CHAPTER 3

### METHODOLOGY



The aims of this chapter are to inform the techniques and methods that have been leveraged to develop the project. The flow of the project is divided into two phases and carefully explained. The reason of choosing each of the method will also be provided.

### 3.1 Research Methodology

The flowchart for this project would consist of several key stages, including data collection and pre-processing, model development, and application development. The flowchart is illustrated below in Figure 3.1.

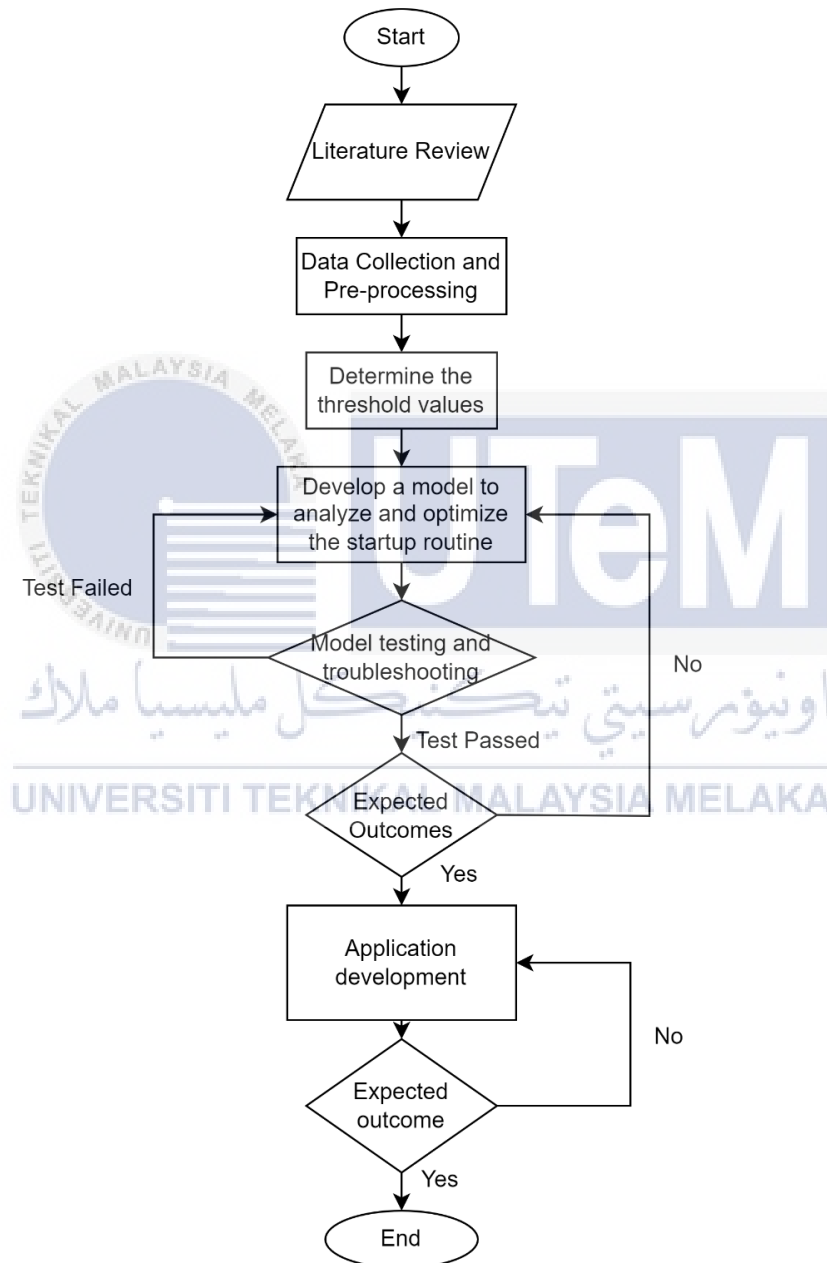


Figure 3.1 Flowchart of Research Methodology

### 3.2 Data Collection

The data collection is done by obtaining data from the collaborating company, Malakoff Corporation Berhad. By gathering the data needed, the underlying pattern of the data and behavior of the plant can be modelled. The collaborating company collected the data by placing sensors strategically across the generator. The sensor selection is also crucial for their ability to capture parameters change across startup timeline.

Collaboration with the company provides valuable insights and access to critical information[21]. The company presents the chain of operation, outlining the sequence of steps required for the power generation process, in a comprehensive Word document. This document serves as a crucial guide, providing a clear understanding of the operational workflow and the interdependencies between different operations. The prerequisite conditions for each step are listed down in great detail, offering a thorough understanding of the necessary requirements for a successful process.

Moreover, the company also provides a detailed document to be used as a guide to understand the datasheet by labelling the sensor data. It enables precise identification and location of the sensor in the plant. By using the guide, the data analysis can be carried out with accurate mapping between the physical sensor to the readings in the Excel dataset.

It is important to highlight that the data collection process relies heavily on the collaboration and partnership with the company due to the sensitive nature of the data. As such, strict confidentiality measures and data security protocols are implemented to safeguard the proprietary information. The collaborative nature of the data

collection process fosters trust and mutual understanding between the research team and the company, ensuring a smooth and productive data collection experience.

Overall, the data collection phase benefits greatly from the collaborative effort with the company. The presented chain of operation, with its detailed prerequisites, guides the data collection process and facilitates the understanding of the power generation system. Additionally, the inventory of sensors and their label names enables precise data collection and mapping of sensor readings to specific components. The partnership with the company ensures the protection of sensitive data, fostering a successful and mutually beneficial collaboration throughout the data collection phase.

The collected data was then meticulously organized and stored in a spreadsheet format using Excel. This approach ensured that the data could be easily accessed, analyzed, and manipulated for subsequent stages of the research. The dataset encompassed a comprehensive range of measurements, including drum level, drum pressure, turbine speed, and temperature readings from various components within the power generator. These parameters were specifically chosen due to their significant impact on assessing the generator's operational efficiency and overall performance. The sample of data extracted from the dataset is shown in Figure 3.2.



Di Time	15-May-20 08:42:00																																									
Declared Available Time Interval	15-May-20 09:44:00 1s	GENERATOR WATTS	HP DRUM LEVEL SETPOINT	HP DRUM LEVEL	HP DRUM PRESSURE	IP DRUM LEVEL	IP DRUM PRESSURE	LP DRUM LEVEL SETPOINT	LP DRUM LEVEL	LP DRUM PRESSURE	TURBINE VOUT TIMER	HP TURBINE SPEED	HP TURBINE SPEED (RPM)	THM_RPM	LBFD	FLAME DETECTED	HP SPEED MINIMUM FIRING	HP SPEED MAXIMUM FIRING	SPEED/STROK RATIO VALUE	SPIN SERVO POSITION	FRAG_INT	FRAG_OUT	SRV GAS CONTROL VALVE SERVO COMMAND	SRV GAS CONTROL VALVE SERVO COMMAND	SRV GAS CONTROL VALVE SERVO COMMAND	SRV GAS CONTROL VALVE SERVO COMMAND	TURBINE INLET GUIDE VALVE SERVO VLV COMMAND	TURBINE INLET GUIDE VALVE SERVO VLV COMMAND	TURBINE WARNING	COMPLETE INCREASE FUEL TEMPERATURE	GT START COMMAND	GT START COMMAND	SPEED REFERENCE	AS_SEQ04 BOOK	GENERATOR BREAKER	STEAM TURBINE METAL TEMPERATURE	HP STEAM HEATER PRESSURE	IP MAIN STEAM PRESSURE	HP STEAM BYPASS CONTROL VALVE I/P CONVERTER	IP MAIN STEAM CONTROL VALVE I/P CONVERTER	STEAM TURBINE PC SELECTED	EX TEMP MEDIAN CORRECTED BY AVERAGE
15-May-20 08:42:00	0.61	-519.19	-445.26	48.08	-388.66	6.87	-269.60	-312.66	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	90.42	FALSE	100.10	FALSE	FALSE	482.80	47.99	6.74	0.00	0.00	FALSE	169.92											
15-May-20 08:42:01	0.61	-519.22	-445.38	48.08	-388.63	6.87	-269.60	-312.82	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.73	FALSE	100.10	FALSE	FALSE	482.90	47.99	6.74	0.00	0.00	FALSE	169.88											
15-May-20 08:42:02	0.61	-519.25	-446.05	48.08	-387.76	6.87	-269.59	-312.86	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.73	FALSE	100.10	FALSE	FALSE	482.84	47.99	6.74	0.00	0.00	FALSE	169.81											
15-May-20 08:42:03	0.61	-519.25	-446.40	48.08	-388.05	6.87	-269.60	-312.04	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.73	FALSE	100.10	FALSE	FALSE	482.84	47.99	6.74	0.00	0.00	FALSE	169.81											
15-May-20 08:42:04	0.61	-519.25	-446.09	48.07	-388.00	6.87	-269.60	-310.11	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.94	FALSE	100.10	FALSE	FALSE	482.79	47.98	6.74	0.00	0.00	FALSE	169.92											
15-May-20 08:42:05	0.61	-519.28	-446.36	48.07	-387.71	6.87	-269.60	-309.25	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.24	FALSE	100.10	FALSE	FALSE	482.87	47.98	6.74	0.00	0.00	FALSE	169.88											
15-May-20 08:42:06	0.61	-519.31	-445.93	48.07	-388.31	6.87	-269.60	-310.15	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.24	FALSE	100.10	FALSE	FALSE	482.85	47.98	6.74	0.00	0.00	FALSE	169.85											
15-May-20 08:42:07	0.61	-519.34	-445.51	48.06	-388.10	6.87	-269.60	-310.89	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.87	FALSE	100.10	FALSE	FALSE	482.89	47.97	6.74	0.00	0.00	FALSE	169.85											
15-May-20 08:42:08	0.61	-519.41	-445.20	48.06	-387.49	6.87	-269.60	-312.08	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.73	FALSE	100.10	FALSE	FALSE	482.80	47.97	6.74	0.00	0.00	FALSE	169.85											
15-May-20 08:42:09	0.61	-519.44	-444.66	48.05	-387.33	6.87	-269.59	-313.44	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.87	FALSE	100.10	FALSE	FALSE	482.85	47.96	6.74	0.00	0.00	FALSE	169.88											
15-May-20 08:42:10	0.61	-519.47	-444.98	48.05	-387.54	6.87	-269.61	-312.49	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.94	FALSE	100.10	FALSE	FALSE	482.80	47.96	6.74	0.00	0.00	FALSE	169.92											
15-May-20 08:42:11	0.61	-519.53	-444.71	48.04	-387.84	6.87	-269.60	-311.47	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.73	FALSE	100.10	FALSE	FALSE	482.79	47.96	6.74	0.00	0.00	FALSE	169.81											
15-May-20 08:42:12	0.61	-519.59	-444.29	48.04	-388.00	6.87	-269.61	-310.52	2.61	FALSE	0.13	4.00	FALSE	FALSE	-100.00	-25.00	-25.00	29.00	FALSE	89.73	FALSE	100.10	FALSE	FALSE	482.85	47.95	6.74	0.00	0.00	FALSE	169.81											

Figure 3.2 Sample Excerpt of Start-Up data provided by the company

Before commencing data collection, thorough calibration procedures were conducted for each sensor. This step was crucial to ensure that the measurements obtained were accurate and reliable. Additionally, regular maintenance and monitoring of the sensors were performed throughout the data collection process to preserve their optimal functioning.

Ethical considerations were strictly adhered to during data collection. This involved obtaining informed consent from relevant stakeholders and ensuring the confidentiality of any sensitive information or proprietary data obtained.

To ensure the integrity and quality of the collected data, various validation techniques were employed. Cross-checking readings from different sensors and comparing them against known reference values were instrumental in identifying and addressing any inconsistencies or outliers within the dataset[22]. By conducting careful scrutiny and, when necessary, additional measurements, the integrity of the data was preserved.

The comprehensive dataset acquired through this rigorous data collection methodology serves as a solid foundation for the subsequent stages of analysis and evaluation within this thesis. It enables a detailed examination of the power generator's operational characteristics, facilitates the identification of potential issues, and paves the way for the formulation of practical recommendations for improvement.

In conclusion, the data collection methodology implemented in this research employed a strategic placement of sensors, meticulous calibration procedures, regular maintenance, and adherence to ethical guidelines. The resulting dataset, stored in an Excel spreadsheet, provides a reliable and accurate representation of critical parameters such as drum level, drum pressure, turbine speed, and component temperatures. This comprehensive dataset forms the basis for further analysis and insights throughout the thesis.

### **3.3 Data Pre-processing**

The data preprocessing phase played a crucial role in refining the collected dataset to ensure its suitability for analysis and optimization purposes. This section outlines the various steps undertaken during the preprocessing stage.

Firstly, the variables need to be categorized into two classes, measured variable and controllable variables. Measured variable includes factors such as atmospheric pressure and surrounding temperature that can only be measured but not optimized. And the controllable variables will be the main subject of the optimization, by carefully examining the variables and their impact on the output power, a subset of essential variables can be retained for further analysis. This is to eliminate any

undesirable noise or redundant information that could lower the performance of the PSO model.

After variable selection, the next step involved converting the Excel spreadsheet containing the collected data into a more standardized format, namely the CSV (Comma-Separated Values) format. Converting to CSV allowed for easier data handling and compatibility with various data analysis tools and software packages. The conversion process ensured that the data maintained its structure and integrity while facilitating seamless integration with subsequent data analysis techniques.

Data cleaning was another crucial aspect of the preprocessing phase. This involved identifying and handling missing data, outliers, and inconsistencies within the dataset. Missing data points were either imputed using appropriate techniques or, if the missingness was significant, carefully analyzed for potential implications on the subsequent analysis and optimization results. Outliers and inconsistencies were identified through statistical methods and domain knowledge, and appropriate treatment strategies were applied, such as removing or adjusting these values based on their impact on the overall dataset. Those outliers and missing values are ignored most of the time in the data analysis.

Additionally, data normalization techniques were implemented to standardize the range and distribution of variables. These techniques are most frequently used to normalize the measured variable because the values that are obtained from the sensors are more than likely not in the standard unit of measurement. These values are usually in the unit of voltage and need to be normalized before included in the model [23]. Normalization ensured that variables with different scales and units were brought to a common scale, allowing for fair comparisons and accurate optimization analysis.

Common normalization techniques used included min-max scaling or z-score normalization, depending on the specific requirements of the optimization algorithms or models employed in the research. Figure 3.3 illustrates the processes of data pre-processing.

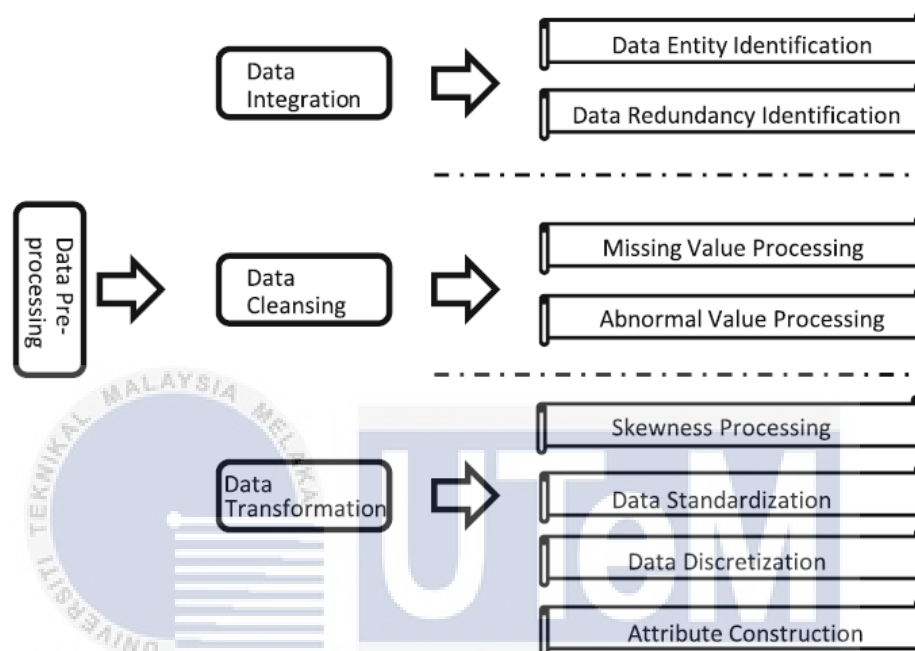


Figure 3.3 Processes of Data Pre-Processing

Throughout the data preprocessing phase, data integrity and quality were maintained by conducting thorough checks and validations. This involved verifying the accuracy of the converted dataset, cross-validating against the original Excel spreadsheet, and ensuring the consistency of variable names and data formats.

In summary, the data preprocessing phase involved several critical steps. This included selecting variables relevant to the optimization process, converting the dataset from Excel to CSV format, handling missing data and outliers, and performing data normalization. These steps were crucial in refining and preparing the dataset for

subsequent analysis and optimization, ensuring the accuracy, consistency, and reliability of the data used in the thesis.

### 3.4 Data Analysis

Data analysis is usually done using simulation model, especially with complex optimization problem like the one this project aims to solve [24]. However, in this project, there will be no simulation process, instead curve fitting will be used to determine the relationship between the dependent and independent variable. This is because there is not enough information to construct a dynamic model as it involves sensitive information that is best left undisclosed. Curve fitting in MATLAB is a valuable technique that can be applied to the project, specifically when working with datasets obtained from an Excel file. MATLAB provides powerful tools and functions for curve fitting, allowing us to analyze and model the relationship between variables in the dataset. By importing the dataset into MATLAB, we can use functions like "xlsread" or "readtable" to extract the data into MATLAB variables. Once the data is available, we can utilize various curve fitting methods, such as linear regression, polynomial fitting, or custom models, to find the best-fit curve that represents the relationship between the variables. MATLAB's curve fitting toolbox offers functions like "fit" and "cftool" that provide easy-to-use interfaces for performing curve fitting and obtaining the fitted curve equation, coefficients, and goodness-of-fit measures. With the fitted curve, we can make predictions, interpolate missing values, or gain insights into the behavior of the variables in the dataset. Curve fitting in MATLAB enhances the project's data analysis capabilities, enabling a deeper understanding of the system dynamics and aiding in decision-making processes.

### 3.5 MATLAB

The coding in MATLAB consists of three parts, the first part implements the PSO according to the formula fitted in the second coding, and the third part is the GUI implemented to provide convenience to the user to compare the startup time with and without the optimization. The provided code implements the Particle Swarm Optimization (PSO) algorithm to optimize the start-up routine parameters of a power generator. The PSO algorithm aims to find the best values for the parameters that minimize the difference between the desired and actual power output over time. The algorithm begins by defining the PSO parameters, search space boundaries, and initializing particles' positions and velocities. It also initializes personal and global best positions and costs based on the initial positions.

The PSO algorithm iteratively updates the particle velocities and positions based on the best positions found so far. It also enforces the search space bounds to ensure that the particles remain within the specified range. The personal and global best positions and costs are updated as better parameters are found. The algorithm continues for a specified number of iterations, tracking the best cost achieved.

After running the PSO algorithm, the code prints the final results, including the best parameters and cost. It then proceeds to plot the desired and actual power output over time using the optimized parameters. The desired power output is defined by a set of power values plotted over the timeline of the start-up sequence, while the actual power output is calculated based on the optimized start-up routine parameters.

Additionally, the code includes a section related to turbine speed and temperature constraints. It defines a start-up routine timetable, sets constraints for the turbine speed and temperature, and formulates an optimization problem. The objective of the

optimization problem is to minimize the overall duration of the start-up routine, subject to the defined constraints. The problem is solved, and the results, including the time saved for each start-up sequence, duration, data set, and change in power over time, are displayed in a GUI application with the use of MATLAB app designer.

In summary, the code uses the PSO algorithm to optimize the start-up routine parameters of a power generator. It iteratively updates particle positions and velocities to find the best parameter values that minimize the difference between the desired and actual power output. The code also considers constraints on the controllable variable such as the fuel supply, drum level setpoint and so on. The optimized parameters are then used to plot the desired and actual power output over time for the sake of comparison with the original dataset and the fitted curve of the original dataset.

### **3.6 Optimization of Start-Up Routine**

#### **3.6.1 Setting of Parameters**

The selection of optimal values for parameters, such as turbine speed, drum pressure, drum level, steam temperature, and others, is a critical aspect of the power generator's operation. These parameters directly influence the readings obtained from sensors and subsequently impact the efficiency and performance of the power generation process. Moreover, their accurate determination is crucial in ensuring the fulfillment of prerequisites for the smooth execution of the start-up routine.

The start-up routine in the power generator involves a chain of interconnected operations, each with specific prerequisites that must be fulfilled before proceeding to the next step. For instance, certain parameters need to reach specified thresholds or

stable levels before the subsequent operation can be initiated. Failure to meet these prerequisites can result in delays or even failure of the start-up process.

To select the best values for the parameters, an iterative optimization approach may be employed. This involves adjusting the parameter values within a given range and evaluating their impact on the fulfilment of prerequisites and subsequent sensor readings. Various optimization algorithms, such as gradient-based methods or evolutionary algorithms, can be utilized to efficiently explore the parameter space and identify the values that optimize the start-up routine.

During the optimization process, it is crucial to consider the interdependencies between parameters and their effects on sensor readings. For example, increasing the turbine speed may influence drum pressure or temperature, which can directly affect the time required to fulfil certain prerequisites. Careful analysis and understanding of these relationships are vital to ensure an effective optimization strategy.

In addition to optimization, the selection of parameter values should take into account operational constraints, safety considerations, and equipment limitations. Parameters should be set within permissible ranges to avoid compromising the stability and integrity of the power generator.

Overall, the selection of optimal parameter values is essential to ensure the successful fulfilment of prerequisites during the start-up routine of the power generator. By employing optimization techniques and considering the interdependencies between parameters and sensor readings, it is possible to enhance the efficiency and reliability of the start-up process. Striking the right balance between



parameter values, prerequisite fulfilment, and operational constraints is crucial for achieving optimal power generation performance.

### 3.6.2 Reducing the Process Time

Efficiently reducing process time is a key objective in power generation, as it leads to enhanced productivity and operational effectiveness. This section focuses on strategies to minimize process time, with specific examples related to purging time, the fuel amount in the ignition chamber, and simultaneous process execution.

Purging time is a crucial step in power generation, which involves removing any unwanted gases or contaminants from the system before initiating full operation. Turbine speed plays a significant role in reducing purging time. By increasing the turbine speed, a higher flow rate of gases and contaminants can be achieved, expediting the purging process and reducing the overall time required.

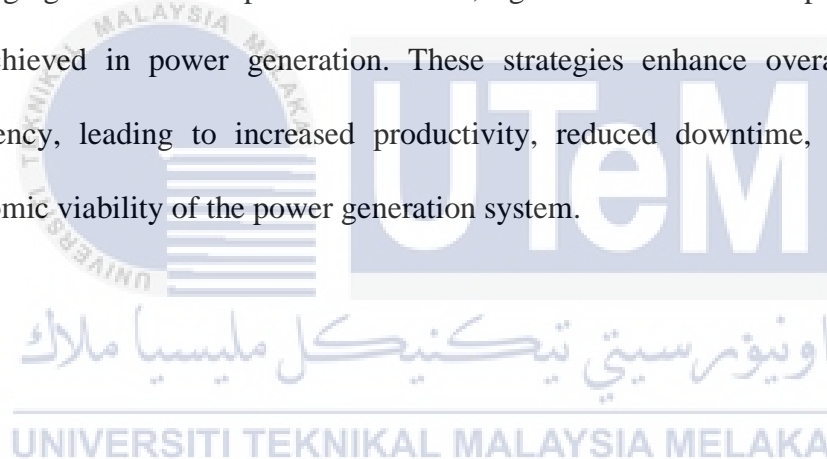
Furthermore, the fuel amount in the ignition chamber is another parameter that influences process time. The fuel amount affects the steam temperature and drum pressure, which are critical for initiating and maintaining the power generation process. Optimizing the fuel amount in the ignition chamber, based on the desired steam temperature and drum pressure, ensures efficient and timely power generation initiation.

In addition to optimizing individual parameters, simultaneous process execution can significantly reduce process time. Certain operations within the power generation system can run concurrently, allowing for parallel execution and expedited completion of the overall process. For example, while the purging process is underway, other

preparatory tasks, such as preheating the steam, can be initiated simultaneously. This parallel execution of processes minimizes idle time and maximizes overall efficiency.

It is important to note that careful consideration should be given to safety and operational constraints when implementing simultaneous process execution. Proper coordination, monitoring, and control mechanisms need to be in place to ensure that the simultaneous processes do not compromise system integrity or result in undesirable consequences.

By strategically optimizing parameters such as turbine speed, fuel amount, and leveraging simultaneous process execution, significant reductions in process time can be achieved in power generation. These strategies enhance overall operational efficiency, leading to increased productivity, reduced downtime, and improved economic viability of the power generation system.



## CHAPTER 4

### RESULTS AND DISCUSSION



This chapter shed some light on the results obtained from the research project and contains discussions regarding the results. A few snippets of the coding and their respective remarks will be provided and given emphasis. And the components of the research project will be dissected and inspect meticulously. The optimized results will be compared to the original data to graphically represent the improvement. The accuracy of the model constructed will be tested and explanation will be provided regarding it. Lastly, the chapter will be concluded with discussions and research limitations.

### 4.1 Result Summary

The block diagram below in Figure 4.1 sums up the results of the project. It consists of four components namely PSO, Data Analysis, Manual Work and Display GUI. Design plan is also included in Figure 4.2.

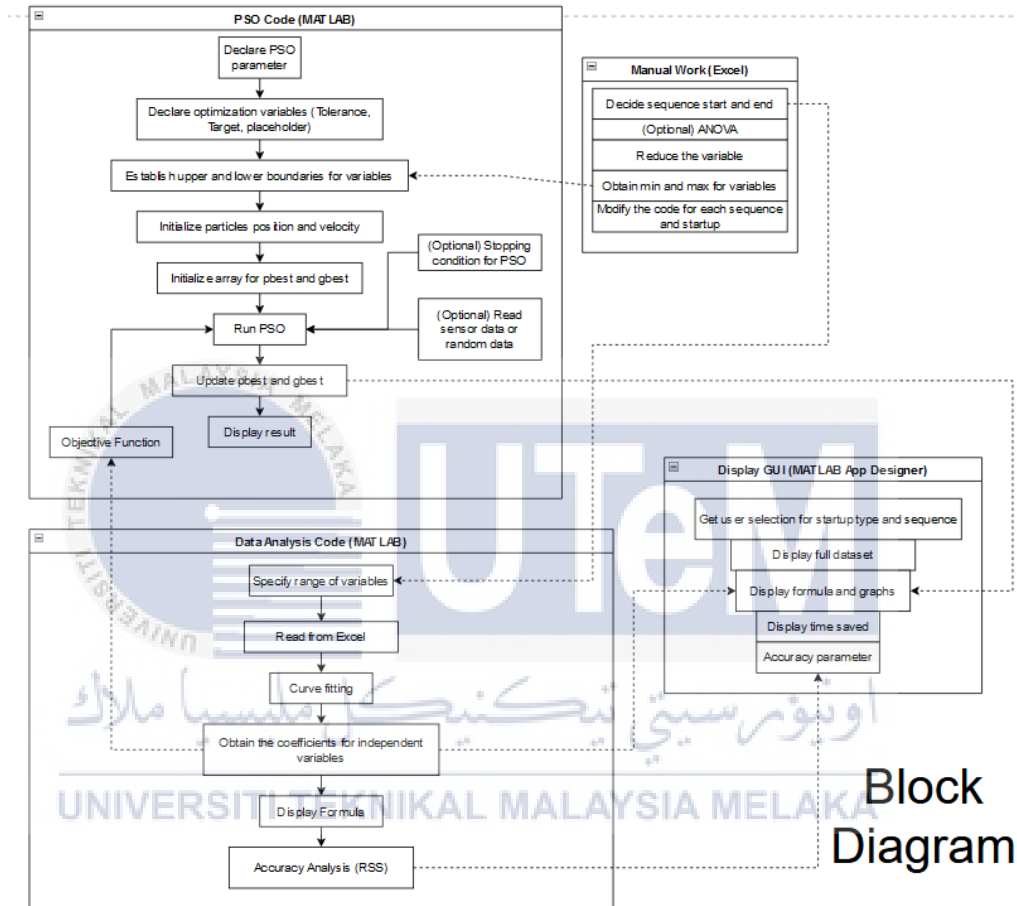


Figure 4.1 Block Diagram

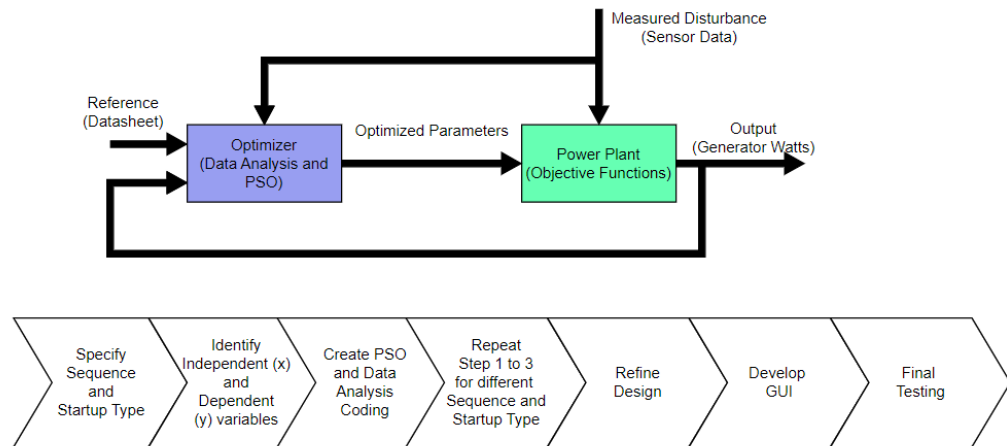


Figure 4.2 Design Plan

## 4.2 Manual Work

At the commence of the project the data that is collected needs to be pre-processed and a lot of it is being done manually. One such procedure is deciding where the sequence start and end. The process cannot be replicated by using code as every sequence have different starting and stopping criteria and MATLAB cannot identify the value of the criteria row by row in the datasheet.

Analysis of Variance (ANOVA) is also being done manually, but it can be automated using `anova()` function in MATLAB. ANOVA produces a table of variables along with their respective p-value which indicates how influential the independent variable is to the dependent variable. ANOVA is not being implemented for all of the sequence because the accuracy will drop after excluding a certain number of variables. Only the initiate sequence of cold startup includes feature selection done by ANOVA and the accuracy experienced a substantial decrease.

Upper and lower boundary will also need to be obtained manually for each sequence. Last but not least, the code will need to be modify manually per sequence and startup type. There is a way to automate the process that is by developing an automation script but that is out of the scope of this project.

### 4.3 Coding Analysis

#### 4.3.1 Particle Swarm Optimization Model

The model is constructed in MATLAB, first the PSO parameters such as  $w$ ,  $c_1$  and  $c_2$  is defined.  $w$  dictates the step size each parameter can take, so the speed of the convergence can be increased with a higher  $w$  value.  $c_1$  and  $c_2$  define the ability of the particle to be influenced by their best personal best position and the global best position respectively.

After that, the optimization parameters are defined the parameters include tolerance, target value and placeholders. The tolerable error in ideal power level is defined as tolerance, it is usually kept under 0.01. Whereas target value is simply the ideal power level. The placeholder to hold values such as the personal and global best position is also declared beforehand.

After the declaration of the parameters above, PSO can be initiate by initializing the particles starting position and velocity. The particles take on a random position within the upper and lower boundary. The velocity is also randomized so that the particles can search within such boundaries.

The PSO then runs until it reached its maximum iteration or until it reached it is the ideal power level. Throughout the process, the personal best and global best position

is updated continuously where the value of the parameter is stored in the array declared beforehand. The position is a set of parameters that differs from one sequence and another. Whether if the position of the particle is desirable is decided by the objective function, the objective function will output a value which is the power level. This PSO differs from other in the aspect of desirable value as it updates the personal and global best with the value closest to ideal power level not the maximum or minimum value. The coding snippet of how this can be achieved is shown below in Figure 4.3.

```

% Update global best position and cost
[min_cost, min_index] = min(pbest_cost);
if abs(min_cost - target_value) < abs(gbest_cost - target_value)
    gbest_cost = min_cost;
    ideal_reached=false;
    if abs(gbest_cost - target_value) <= 0.1 && ideal_reached==false
        IdealIter=iter;
        ideal_reached=true;
    end
    gbest_x = pbest_x(min_index, :);
end
end

```

Figure 4.3 Updating with Value Closest to Target Value and One-Time Nested Condition Checking Loop

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Last but not least, the result of each iteration is displayed and the ideal set of parameters for each iteration is shown. The output of the coding is a time series set of parameters that can achieve the power level associated with it. With an example of it displayed in Figure 4.4.

```

Command Window
-749.2800 -609.4918 -290.4334 -336.5800 -278.7372 0.1300 4.0000 -150.3655 103.5700 31.5364 29.8100 0 0 30.8311
Columns 15 through 16
55.6200 752.9100
Iteration 46: Best cost = -1.2312
Columns 1 through 14
-749.2800 -609.4918 -290.4334 -336.5800 -278.7372 0.1300 4.0000 -150.3655 103.5700 31.5364 29.8100 0 0 30.8311
Columns 15 through 16
55.6200 752.9100
Iteration 47: Best cost = -1.2312
Columns 1 through 14
-749.2800 -609.4918 -290.4334 -336.5800 -278.7372 0.1300 4.0000 -150.3655 103.5700 31.5364 29.8100 0 0 30.8311
Columns 15 through 16
55.6200 752.9100
Iteration 48: Best cost = -1.2312
Columns 1 through 14
-749.2800 -609.4918 -290.4334 -336.5800 -278.7372 0.1300 4.0000 -150.3655 103.5700 31.5364 29.8100 0 0 30.8311
Columns 15 through 16
55.6200 752.9100
Iteration 49: Best cost = -1.2312
Columns 1 through 14
-749.2800 -609.4918 -290.4334 -336.5800 -278.7372 0.1300 4.0000 -150.3655 103.5700 31.5364 29.8100 0 0 30.8311
Columns 15 through 16
55.6200 752.9100
Iteration 50: Best cost = -1.2312
Columns 1 through 14
-749.2800 -609.4918 -290.4334 -336.5800 -278.7372 0.1300 4.0000 -150.3655 103.5700 31.5364 29.8100 0 0 30.8311
Columns 15 through 16
55.6200 752.9100

```

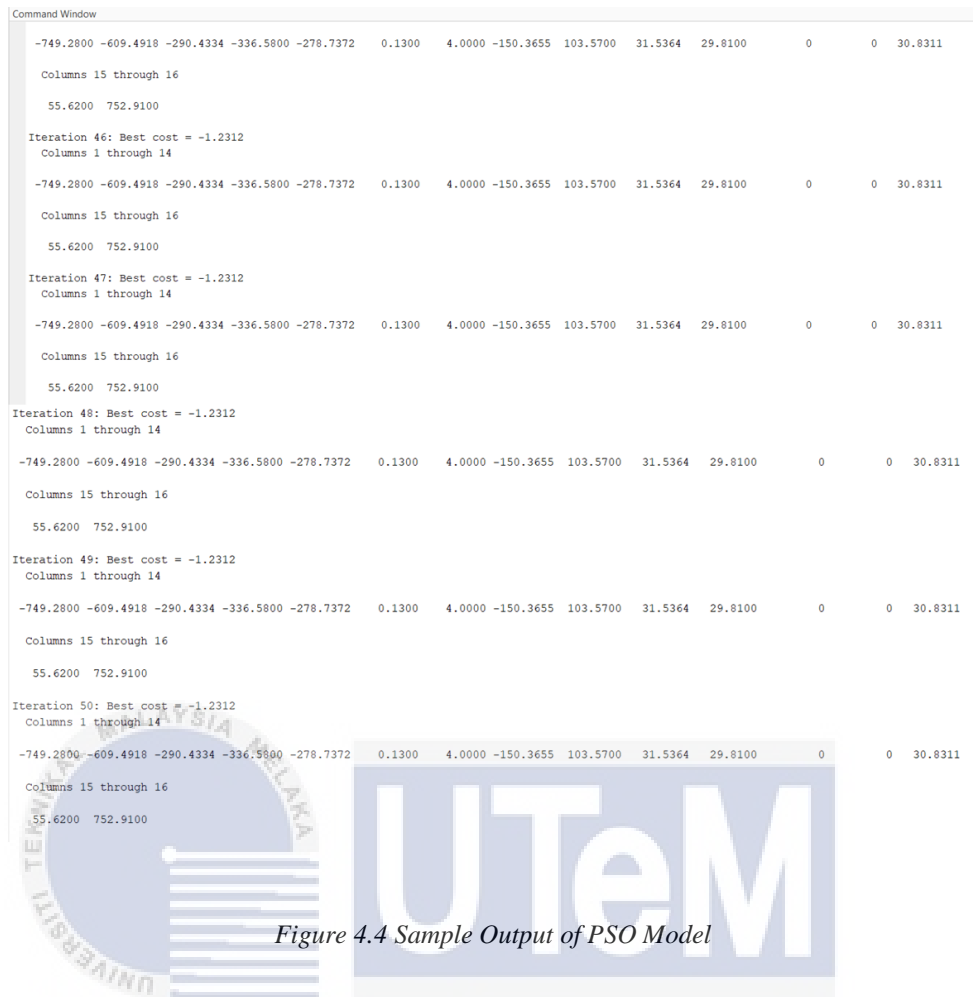


Figure 4.4 Sample Output of PSO Model

The time taken for the particle to converge to the power level differs each time after running the code as the particles' position and velocity is randomized, but the graph representing the curve is the same with marginal difference.

### 4.3.2 Data Analysis

The data analysis code is the shortest among the three, but it is the first process that the receive the data in the form of Excel datasheet. The code takes in the range of independent variables, dependent variable and the total data point. According to the range provided, it reads into the Excel file and then performs curve fitting using the



function `polyfitn()` before plotting the data using `polyvaln()`. The section of the code is showcase below with their respective input parameters in Figure 4.5.

```
% Fit the polynomial
coefficients = polyfitn(independent, dependent, degree);

% Evaluate the polynomial fit
yFit = polyvaln(coefficients, independent);
```

*Figure 4.5 Curve Fitting*

The coefficient of each variable will be determined by the code and stored inside an array. An output with the form of a formula will be the final product of this code. It also has a graph display function, but that function is later moved to the GUI section to centralize all the display components.

### 4.3.3 Graphical User Interface (GUI)

The GUI provided a means for user to interact with the program written in MATLAB. The GUI is coded with the help of MATLAB App Designer, it is an app designer that can be use with ease as UI component can be constructed by simply dragging the component on the UI view.

The GUI constructed enables users to display the dataset that is associate with a type of start up by users, choose the sequence of the startup, generate a polynomial formula based on the users' selection, subsequently display the comparison between original data, fitted curve and the optimized curve and lastly show the time saved for the startup sequence in selection.

Button is used in the UI to enable users to select the startup type and sequence. Text/Number area and UIAxes are used to display texts/numbers and plot graphs respectively. A snippet of GUI code is provided in Figure 4.6.

```
% Selection changed function: SequenceButtonGroup
function SequenceButtonGroupSelectionChanged(app, event)
    selectedButton = app.SequenceButtonGroup.SelectedObject;
    if app.PurgeButton.Value==1 && app.ColdButton.Value==1
        Purge_Cold_PSO;
        Purge_Cold;
        cla(app.UIAxes);
        formula = coefficientString;
        app.TextArea.Value = formula;
        app.TimeSavesdEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y,'--or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted Curve');
    elseif app.InitiateButton.Value==1 && app.ColdButton.Value==1
```

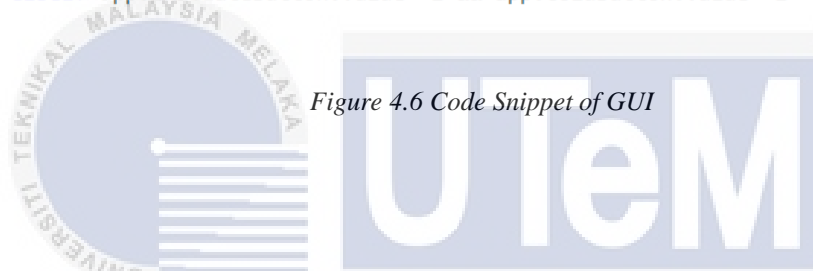


Figure 4.6 Code Snippet of GUI

Every time the user selects a startup type and sequence, PSO model and data analysis coding will execute sequentially and produce formula, optimized curve, original curve, fitted curve, time saved, accuracy metrics and dataset accordingly. The result of the display GUI will also be the final output of this project.

#### 4.4 Sensor Integration

The project also includes an optional procedure that integrates sensor to the MATLAB coding in order to simulate the natural fluctuation of measured variables such as surrounding temperature, atmospheric pressure, humidity and such. An interface in the form of an Arduino board will first need to be procured and connected to both the sensor and Personal Computer correctly. After that, the code in PSO will

need to be modify accordingly for it to be include inside the PSO iteration. In Figure 4.7, a snippet of the coding is provided below as a reference.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create Arduino object
%a = arduino('COM7', 'Uno', 'Libraries', 'I2C');

% Create MPU6050 object
%mpu = mpu6050(a, 'I2CAddress', 0x68);

rray=zeros(num_particles,3);

```

*Figure 4.7 Declaration of Arduino Objects*

The section of code creates an Arduino object by specifying the port that is used to establish connection with PC, model of the board, library and the type of connection. Then the MPU6050 Accelerometer object declare the address of the board once it is connected. After that an array is declared to store the readings from the sensor. The array named rray here is an array that has number of particles as rows and three columns.

```

accelData = readAcceleration(mpu);
randData = lb(7) + (ub(7) - lb(7)) *rand(1,1);
mappedValue = 4 + ((accelData - (-20)) / (20 - (-20))) * (703.25 - 4);
rray(i,1)=randData;

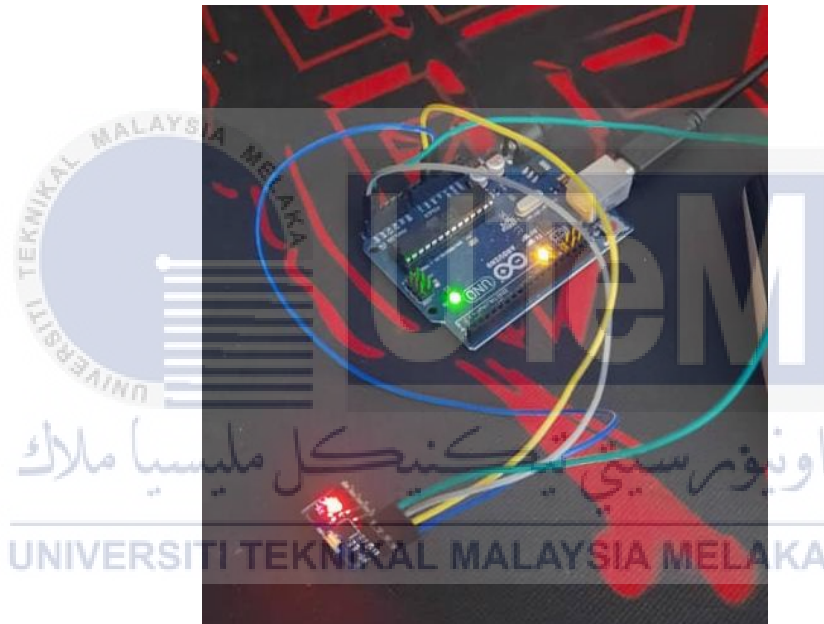
```

*Figure 4.8 Sensor Value Mapping*

After that the value that is obtained directly from the sensors will need to be mapped to the actual value represented. This is because most of the sensor value is send as electrical signal in the form of voltage. The formula of the mapping is given above

with upper boundary and lower boundary as input, and it will return a mapped value that represent the value of that particular variable.

The main purpose of sensor integration is to simulate the changes in uncontrollable variable and a prototype constructed with accelerometer is tested out in the research. Besides, the sensor integration is also aimed to prove that sensor can be used to read measured variable to increase the functionality of the project. Said construction of the setup is included below in Figure 4.9.



*Figure 4.9 Construction of Sensor Integration with Accelerometer and Arduino UNO*

#### **4.5 Output Interpretation**

Figure 4.10 below shows the output of the PSO coding or more specifically the output of the PSO coding in the display GUI. The formula is highlighted in grey, it is part of the result of the curve fitting in data analysis. The number of variables in the formula differs for each startup type and sequence so the terms of the formula vary

from one formula to another. The area highlighted in yellow is user selection for sequence and startup type. Noted only one selection is available for each sequence and startup type. The area in black is the unprocessed dataset of each startup type and the one in red is all the curves. The most key component of the GUI is the time saved of each sequence and it is represented in green box. It is calculated by subtracting time taken when ideal power level is reached from total time taken.

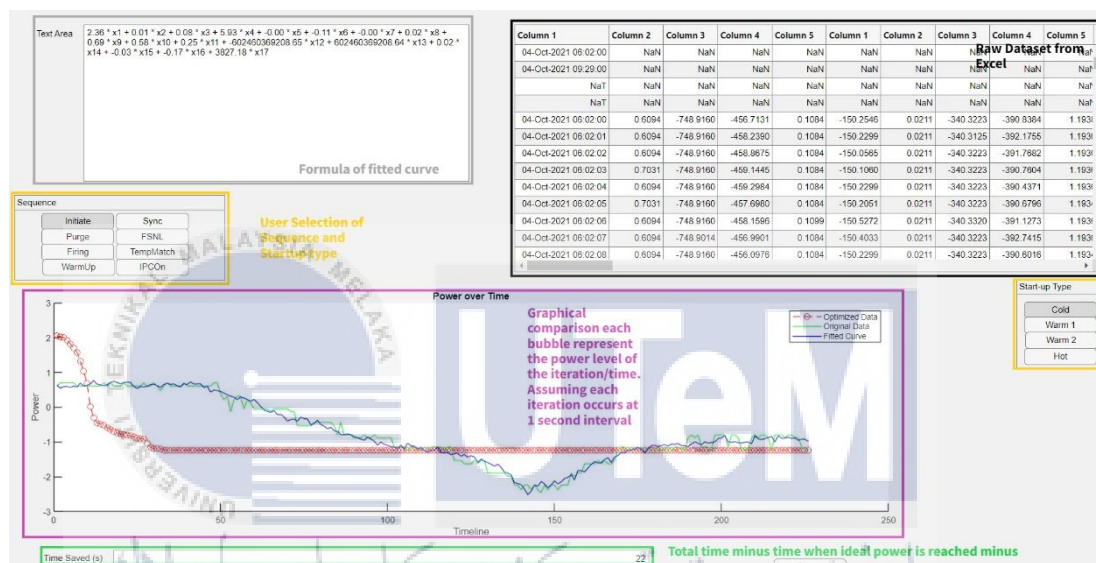


Figure 4.10 Output with Highlighted Explanation

The accuracy metrics is added later to the GUI to show how accurate the fitted curve is with the original data. The display area of the metrics is positioned below the startup type selection. A sample of the section is illustrated below in Figure 4.11.



Figure 4.11 Accuracy Metrics

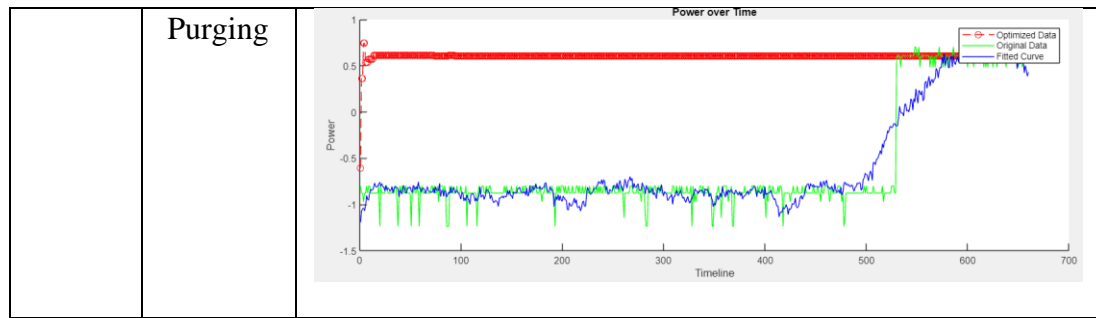
#### 4.6 Comparison Between Optimized and Unoptimized Start-Up Time

For sake of comparison, two identical sequences from each startup type are chosen and displayed in Table 1.

Table 4.1: Output Curves for Initiate and Purging Sequence

Startup Type	Sequence	Graphs
Cold	Initiate	
	Purging	

<p>Warm 1</p>	<p>Initiate</p>	
	<p>Purging</p>	
<p>Warm 2</p>	<p>Initiate</p>	
	<p>Purging</p>	
<p>Hot</p>	<p>Initiate</p>	



#### 4.7 Time Saved

Time is the chosen metric for the performance measurement for the PSO model to investigate the speed of convergence. It is calculated by subtracting the time taken to reach ideal power level from total time taken for the unoptimized curve. The PSO is proven to accelerate the process of a chosen sequence ranging from 14.2 percent to as high as 78.4 percent depending on the startup type and sequence as well as the PSO parameters such as  $w$ ,  $c1$ ,  $c2$  and number of particles. Five samples are taken for each startup type and sequence because the time saved will vary each time running the PSO as it is highly dependent on the randomness of the initial particle position and velocity. The time saved for each sequence is shown below in Table 2.

**Table 4.2: Timesaved for Each Sequence**

Startup Type	Sequence	Time Saved/s	Average Time Saved	Time Taken/s (Unoptimized)	Percentage of Time Saved/%
Cold	Initiate	5, 185, 190, 191, 201	154.4	226	68.3



	Purging	437, 401, 552, 259, 469	423.6	660	64.2
Warm 1	Initiate	136, 155, 152, 153, 192	157.6	201	78.4
	Purging	348, 461, 326, 411, 451	399.4	661	60.4
Warm 2	Initiate	37, 41, 35, 1, 42	31.2	220	14.2
	Purging	169, 204, 123, 371, 356	244.6	660	37.1
Hot	Initiate	53, 183, 77, 162, 132	121.4	239	50.8
	Purging	406, 316, 369, 285, 383	351.8	660	53.3

#### 4.8 Impact of PSO Parameters on Optimized Curve

This section demonstrates the influence PSO parameters have on the optimized curve. Observation can be made where the larger the number of particles the faster the

PSO will reach convergence. Besides, the higher  $w$  value also reduces the time needed for convergence [25]. Whereas  $c_1$  and  $c_2$  are kept constant with the same value of 1.5 as it dictates how likely the particle will follow personal best and global best position respectively. This way the particle will follow both positions equally. From Figure 4.12 to Figure 4.17 the output curves of the PSO model is shown with varying optimizing parameters.



Figure 4.12  $num\_particles = 100, w = 0.7, c_1, c_2 = 1.5$

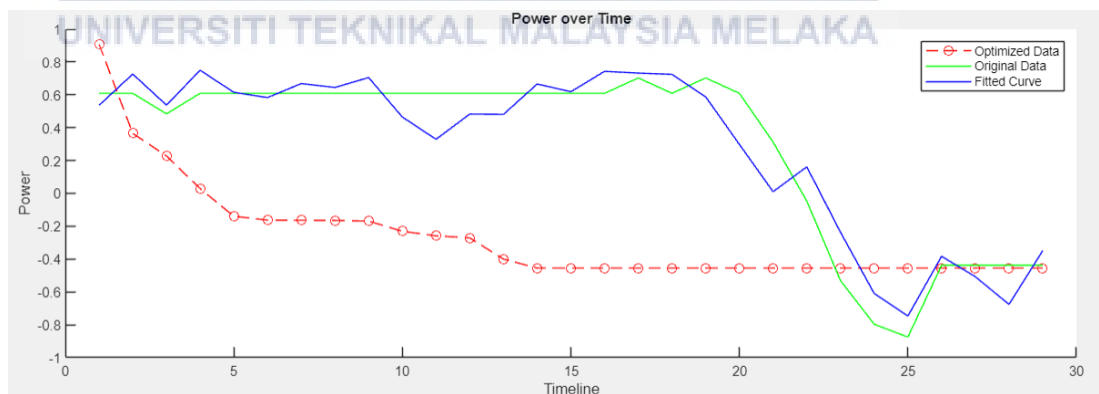


Figure 4.13  $num\_particles = 50, w = 0.7, c_1, c_2 = 1.5$

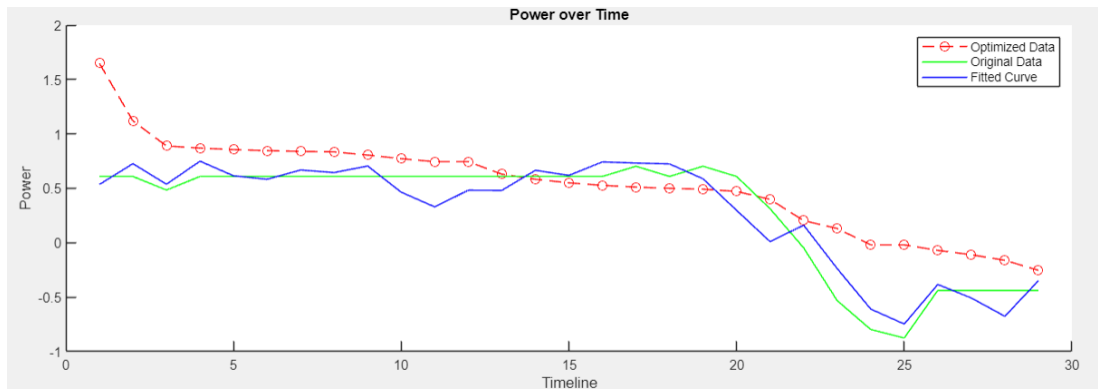


Figure 4.14  $num\_particles = 10$ ,  $w = 0.7$ ,  $c1, c2 = 1.5$



Figure 4.15  $num\_particles = 50$ ,  $w = 0.5$ ,  $c1, c2 = 1.5$

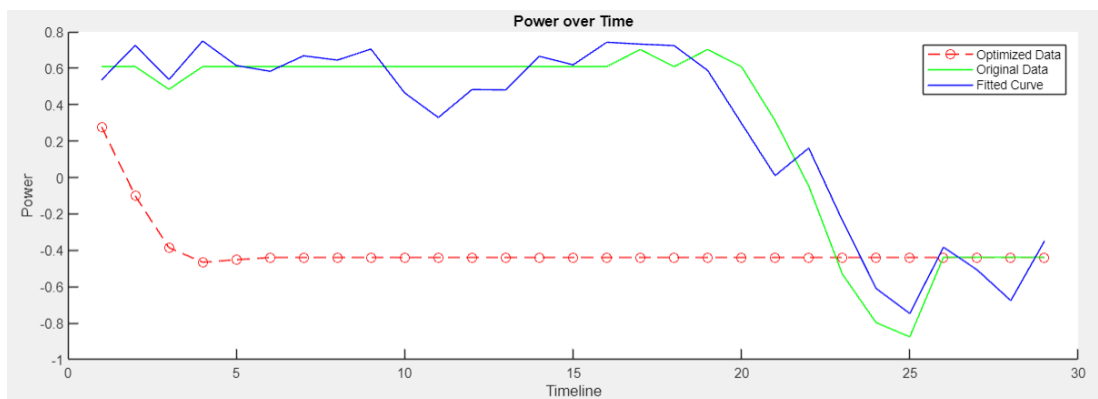


Figure 4.16  $num\_particles = 50$ ,  $w = 0.9$ ,  $c1, c2 = 1.5$

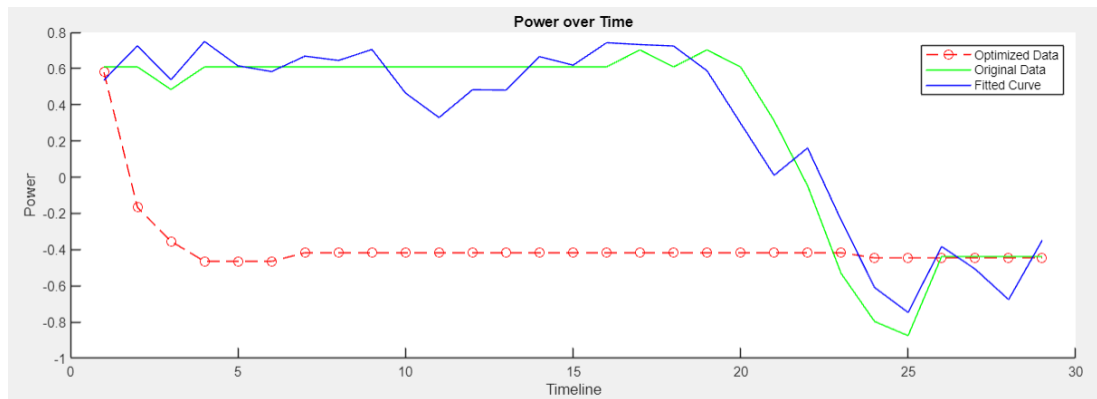


Figure 4.17  $num\_particles = 50$ ,  $w = 1.0$ ,  $c1, c2 = 1.5$

## 4.9 Accuracy of the Particle Swarm Model

Two crucial metrics are chosen to measure the accuracy of the formula obtained. It is of utmost importance that the formulas obtained have high accuracy because it is the objective function of the PSO model. The two metrics are Residual Sum of Squares and Mean Error. One important finding that is being discovered in the process of accuracy measurement is the dropping of accuracy when reducing number of variables of objective function as a result of variable selection [26]. By excluding some variables, the behavior of the power plant cannot be represented as accurately.

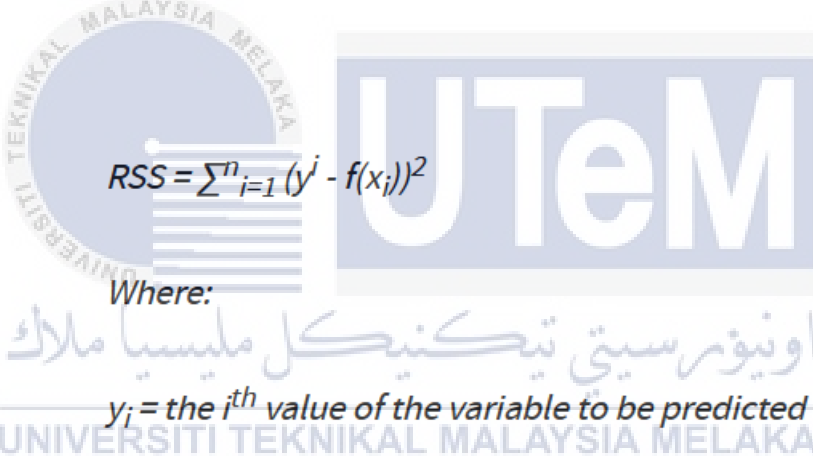
### 4.9.1 Residual Sum of Squares (RSS)

RSS measures the variance in the error of a regression model [27]. It is the measure of difference between the data and the estimation model.

In this project Residual Sum of Squares is used to indicate the accuracy of the objective functions obtained from the curve fitting. The objective function represents the behavior of the power plant at the selected sequence and startup type, thus it is

important that there is a method to measure the goodness of the formula. RSS is applied to the curve fitted from the MATLAB coding. The reason as to why the RSS is not performed on the optimized curve is because the curve is optimized to reduce the time taken for the startup, so naturally it will differ from the original curve by a great margin. Therefore, the correct way is to compare the fitted curve to the actual curve as the formula of the fitted curve is also used to obtain the optimized curve.

The RSS calculation is repeated for each sequence for all the startup type. The formula for RSS is provided in Figure 4.18 and the RSS for each sequence is shown in Table 3.



$$RSS = \sum_{i=1}^n (y_i - f(x_i))^2$$

Where:

$y_i$  = the  $i^{th}$  value of the variable to be predicted

$f(x_i)$  = predicted value of  $y_i$

$n$  = upper limit of summation

Figure 4.18 Formula of RSS Calculation

**Table 4.3: RSS of Each Sequence**

Startup	Sequence	RSS
Cold	Initiate	RSS 3.919
	Purging	RSS 2.566e-20
Warm 1	Initiate	RSS 1.616e-19
	Purging	RSS 2.955e-20
Warm 2	Initiate	RSS 6.626e-21
	Purging	RSS 5.061e-22
Hot	Initiate	RSS 1.221e-20
	Purging	RSS 3.474e-20

#### 4.9.2 Mean Error

Mean error is chosen to represent the deviation of the fitted curve from the original curve obtained from the Excel datasheet. Lesser mean error indicates a less deviation from the actual data point [28]. It is the most direct method of accuracy measurement calculated by obtaining the difference between the predicted and actual data point and averaging the values. The mean error is shown below in Table 4.

**Table 4.4: Mean Error of Each Sequence**

Startup	Sequence	Mean Error
Cold	Initiate	Mean Error -0.00876
	Purging	Mean Error 2.427e-13

Warm 1	Initiate	Mean Error	-2e-12
	Purging	Mean Error	2.601e-13
Warm 2	Initiate	Mean Error	3.7e-13
	Purging	Mean Error	3.408e-14
Hot	Initiate	Mean Error	-4.624e-13
	Purging	Mean Error	-2.824e-13

Noted that the mean error of Initiate sequence from cold startup is relatively high compared to the other sequence because ANOVA is performed as a means of feature selection before curve fitting so the number of variables is reduced so does the accuracy.

#### 4.10 Discussions

The result display in the GUI indicates that the method used to reduce the startup time is valid. This can be observed by the low RSS and Mean Error value. The time saved varies from one sequence to another, but all shows positive improvement. The flaw of this method is that the particles move too quickly to the ideal power level resulting in a premature convergence. Though the approximate behavior of the plant can be summarized in the form of objective function with marginal deviation. Implementation of a way to delay the time of convergence is advisable, but the selection of method must be cautiously carried out. This is because a simple wait time that is introduced artificially is not desirable as the power plant have a certain chain of operation that have dependency on the previous operation.

This brings us to the limitation of the research project, the dataset sourced from Malakoff Corporation Berhad is not sufficient to derive useful detail for feature selection as most of it is highly confidential and sensitive. And there is no way to know the dependency of each parameter simply from the dataset. Besides, the project employs PSO as method of optimization which cannot capture the full complexity of time series problem and sensitivity to initial conditions.

In the research, the chosen method for data pre-processing is manual work, this method is very time consuming as the dataset needs to be modify repeatedly for each sequence and there are thirty-two sequences in total. A revision on the method is advisable to avoid wasting time, and the proposed method is an automated script to perform data splitting according to parameter values, feature selection and procuring maximum and minimum for each variable. Besides, an automated script like PowerShell for windows or AppleScript for apple can be developed to automate the process of code modification in MATLAB.

While this project is still not mature to be used to improve power plant startup time completely, it is deployable as an auxiliary tool to aid the existing system and provide insight into the parameters that is needed to achieve ideal power output for a power plant.

#### **4.11 Research Limitation**

The first limitation is the particles converge too quickly to the objective which in this case is the ideal power level for each sequence. The particles will always move to the objective on the start of the sequence[29], it is not desirable because it does not



represent true behavior of the power plant. The proposed solution to this limitation is to add sort of a time delay or set a constraint on the rate of change of each parameter.

Besides, the time needed for particles to converge is subjected to factors such as sensor involvement, number of particles and max iteration. There is currently no way to speed up this process as the communication between the sensor and the MATLAB code requires a set amount of time.

Generalization issues with PSO include sensitivity to initial conditions, early convergence tendency, and difficulties with adapting to time series problem. Its efficacy varies depending on the type of optimization problem, and its complexity is increased by the requirement for meticulous parameter fine tuning. Constrained generalizability of PSO is further exacerbated by its memory constraints, constraint handling challenges, and restricted applicability in noisy or stochastic situations. Even though PSO has shown to be successful for some optimization tasks, its resilience in a variety of problem cases is still an issue[30]. For this reason, different optimization algorithms should be taken into consideration depending on the particulars of the problem at hand. One such example for time series problem includes Autoregressive Integrated Moving Average (ARIMA) Model.

#### **4.12 Research Sustainability**

This research coincides with Goal 7: Affordable and clean energy, Goal 9: Industry, Innovation and Infrastructure and Goal 13: Climate action. The project does so by providing a means to reduce greenhouse gas emission with improved startup time thus providing clean and affordable energy. It also signifies innovation as the project offers

a novel approach to increase startup efficiency. Lesser greenhouse gas emission also contributes in fighting climate change as lower emission will reduce the atmospheric temperature of the earth.



## CHAPTER 5

### CONCLUSION AND FUTURE WORKS



The last chapter concludes the thesis, it contains conclusion and future works. This chapters summarize the project and provide insight as to where is the future of the research leads. The section future works re-emphasize the flaws and limitations of the project and provide advice on solutions for improvement.

## 5.1 Conclusion

In this thesis I have explored the possibility of leveraging Particle Swarm Optimization to reduce the startup time for power plant startup routine and carried out transient state analysis with MATLAB as tool. I have demonstrated that the startup time can indeed be improved by proposing a procedure of mitigative action in the form of a set of power plant parameters procured from PSO. By doing this, the PSO basically reverse the clock and allow more time for fine tuning of parameters such as signal arc and phase crucial for synchronization to take place. Collection and analysis of data is also being accomplished.

This work has contributed new insight to propose solution for time series problems which in this case is the startup time optimization. Application of PSO to this specific problem is a first so there is a new horizon to discover. Although the method had its weakness of converging too quickly but the potential for real-world implementation is still high regardless.

Future research could further explore the implications of these findings and continue on refine the research to minimize the weakness aforementioned. With this, the thesis has come to a conclude where all the objectives are fulfilled.

## 5.2 Future Works

After the conclusion of thesis, much discovery has been made and with that comes the implications of limitations regarding the research project. Suggestions as to where the future of the research lies can thus be advised.

If this project is to be refined, there are a few key takeaways that can be derived from project limitations. First, the timing of the particle convergence must take into account the parameter dependencies of the power plant. This can ensure that maximum amount of detail can be modelled into the PSO.

Besides, a new script can be developed to automate the tedious manual work that have been done in this research is advisable and the same script needs to be applicable for all of the sequence for ease of use. This way the data pre-processing phase can be sped up by a great margin. The form of the script should be written in PowerShell for Windows or AppleScript for Apple devices, anything similar can also be utilized.

Last but not least, an additional section of GUI can be added to compile the result from all of the sequence to summarize how much the total time saved. The findings have opened up new avenues for future work, promising exciting developments and deeper understanding. This project is still immature and requires a great amount of time to realize its real potential, it would be great if future research can heed the suggestion stated in this section.

## REFERENCES

- [1] W.-P. Schill, M. Pahle, and C. Gambardella, "On Start-Up Costs of Thermal Power Plants in Markets with Increasing Shares of Fluctuating Renewables," *SSRN Electronic Journal*, 2016, doi: 10.2139/ssrn.2723897.
- [2] A. Ghulomzoda *et al.*, "Recloser-based decentralized control of the grid with distributed generation in the Lahsh district of the Rasht grid in Tajikistan, central Asia," *Energies (Basel)*, vol. 13, no. 14, 2020, doi: 10.3390/en13143673.
- [3] Y. Yoshida, T. Yoshida, Y. Enomoto, N. Osaki, Y. Nagahama, and Y. Tsuge, "Start-up optimization of combined cycle power plants: A field test in a commercial power plant," *J Eng Gas Turbine Power*, vol. 141, no. 3, 2019, doi: 10.1115/1.4041521.
- [4] C. Yin, H. Wu, M. Sechilariu, and F. Locment, "Power management strategy for an autonomous DC microgrid," *Applied Sciences (Switzerland)*, vol. 10, no. 11, 2018, doi: 10.3390/app8112202.

- [5] M. Shirakawa, M. Nakamoto, and S. Hosaka, "Dynamic simulation and optimization of start-up processes in combined cycle power plants," *JSME International Journal, Series B: Fluids and Thermal Engineering*, vol. 48, no. 1, 2005, doi: 10.1299/jsmeb.48.122.
- [6] WETO, "How Do Wind Turbines Work?," WETO.
- [7] M. D. P. Buitrago-Villada, S. Garcia-Marin, J. E. Zuluaga-Orozco, and C. E. Murillo-Sanchez, "On the Importance of using an AC or DC Network Model in the Multi-Period Secure Stochastic Optimal Power Flow for Settling a Multidimensional Day-Ahead Market," *IEEE Latin America Transactions*, vol. 19, no. 12, 2021, doi: 10.1109/TLA.2021.9480141.
- [8] M. Goodarzi, "Energy and exergy analyses of a new atmospheric regenerative Brayton and Inverse Brayton cycle," *Energy Reports*, vol. 7, 2021, doi: 10.1016/j.egy.2021.07.034.
- [9] T. k. Ibrahim *et al.*, "The optimum performance of the combined cycle power plant: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 79, 2017. doi: 10.1016/j.rser.2017.05.060.
- [10] W. P. Schill, M. Pahle, and C. Gambardella, "Start-up costs of thermal power plants in markets with increasing shares of variable renewable generation," *Nat Energy*, vol. 2, no. 6, 2017, doi: 10.1038/nenergy.2017.50.
- [11] A. J. Seebregts, "Gas-Fired Power," *IEA ETSAP - Technology Brief E02 – April 2010*, no. April, 2010.

- [12] Z. Geng *et al.*, “Environmental economic dispatch towards multiple emissions control coordination considering a variety of clean generation technologies,” in *IEEE Power and Energy Society General Meeting*, 2015. doi: 10.1109/PESGM.2015.7286311.
- [13] Z. Wu, W. Xu, C. Li, and X. Meng, “A new approach for generator startup sequence online decision making with a heuristic search algorithm and graph theory,” *Energy Reports*, vol. 8, 2022, doi: 10.1016/j.egyr.2022.02.239.
- [14] D. Chao and S. Yongjian, “Working process of steam turbine and establishment of start-up model,” *International Journal of Physics Research and Applications*, vol. 4, no. 1, 2021, doi: 10.29328/journal.ijpra.1001040.
- [15] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, “Particle Swarm Optimization: A Comprehensive Survey,” *IEEE Access*, vol. 10, pp. 10031–10061, 2022, doi: 10.1109/access.2022.3142859.
- [16] A. Sajadi, R. W. Kenyon, and B.-M. Hodge, “Synchronization in electric power networks with inherent heterogeneity up to 100% inverter-based renewable generation,” *Nat Commun*, vol. 13, no. 1, p. 2490, May 2022, doi: 10.1038/s41467-022-30164-3.
- [17] E. Andersson, “Development of a dynamic model for start-up optimization of coal-fired power plants,” *MASTERS THESIS in Automatic Control*, no. June, 2013.



- [18] A. Nannarone and S. A. Klein, "Start-Up Optimization of a CCGT Power Station Using Model-Based Gas Turbine Control," *J Eng Gas Turbine Power*, vol. 141, no. 4, 2018, doi: 10.1115/1.4041273.
- [19] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3142859.
- [20] M. Hübel *et al.*, "Modelling and simulation of a coal-fired power plant for start-up optimisation," *Appl Energy*, vol. 208, 2017, doi: 10.1016/j.apenergy.2017.10.033.
- [21] H. Taherdoost, "Data Collection Methods and Tools for Research; A Step-by-Step Guide to Choose Data Collection Technique for Academic and Business Research," *International Journal of Academic Research in Management (IJARM)*, vol. 2021, no. 1, 2021.
- [22] Y. Roh, G. Heo, and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, 2021. doi: 10.1109/TKDE.2019.2946162.
- [23] Z. Jin, M. Yao, and D. Tao, "Sensor Data Normalization Among Heterogeneous Smartphones for Implicit Authentication," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022. doi: 10.1007/978-3-030-95391-1\_21.

- [24] Y. Zhang, T. Huang, and E. F. Bompard, "Big data analytics in smart grids: a review," *Energy Informatics*, vol. 1, no. 1, 2018, doi: 10.1186/s42162-018-0007-5.
- [25] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing Journal*, vol. 11, no. 4, 2011, doi: 10.1016/j.asoc.2011.01.037.
- [26] M. Krystek and M. Anton, "A weighted total least-squares algorithm for fitting a straight line," *Meas Sci Technol*, vol. 18, no. 11, 2007, doi: 10.1088/0957-0233/18/11/025.
- [27] J. A. Morgan and J. F. Tatar, "Calculation of the Residual Sum of Squares for all Possible Regressions," *Technometrics*, vol. 14, no. 2, 1972, doi: 10.1080/00401706.1972.10488918.
- [28] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean Absolute Percentage Error for regression models," *Neurocomputing*, vol. 192, 2016, doi: 10.1016/j.neucom.2015.12.114.
- [29] Z. Yuan, L. Yang, Y. Wu, L. Liao, and G. Li, "Chaotic particle swarm optimization algorithm for traveling salesman problem," in *Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2007*, 2007. doi: 10.1109/ICAL.2007.4338736.
- [30] A. M. Abdelbar, S. Abdelshahid, and D. C. Wunsch, "Fuzzy PSO: A generalization of particle swarm optimization," in *Proceedings of the*

*International Joint Conference on Neural Networks*, 2005. doi:  
10.1109/IJCNN.2005.1556004.



## APPENDICES

### Appendix A: PSO Sample Coding for Firing Sequence of Cold Startup

```

%clear all
clc

% PSO parameters
num_particles = 50;
max_iterations = 29;
w = 1;
c1 = 1.5;
c2 = 1.5;
target_value = -0.44;
IdealIter=0;

% Search space bounds
lb = [-749.43 -666.99      -470.77      -337.64      -327.18      13.20
      396.25 -127.33      103.43 44.62  29.72  29.98  33.47  5.60      88.31
      74.03  33.06  29.84  61.05];
ub = [-749.41 -663.67      -468.91      -337.60      -321.35      14.12
      423.75 -125.17      103.54 45.42  29.78  30.03  33.53  9.24      88.51
      74.41  33.09  30.16  63.60];
VarCount=numel(ub);

% Initialize particles
x = zeros(num_particles, VarCount);
v = zeros(num_particles, VarCount);
for i = 1:num_particles
    x(i, :) = [-749.43 -665.58 -469.70 -337.60 -323.90 14.12 423.75 -
125.17 103.51 45.42 29.77 30.01 33.51 9.24 88.44 74.10 33.09 30.16
61.53];%lb + (ub - lb) .* rand(1, 10);
    v(i, :) = -1 + 2 .* rand(1, VarCount);
end

% Initialize personal best positions and costs
pbest_x = x;
pbest_cost = zeros(num_particles, 1);
for i = 1:num_particles
    pbest_cost(i) = objective(x(i, :));
end

% Initialize global best position and cost
[gbest_cost, gbest_index] = min(pbest_cost);
gbest_x = pbest_x(gbest_index, :);

% Run PSO

```

```

for iter = 1:max_iterations
    % Update particle velocities and positions
    for i = 1:num_particles
        r1 = rand(1, VarCount);
        r2 = rand(1, VarCount);
        v(i, :) = w * v(i, :) + c1 .* r1 .* (pbest_x(i, :) - x(i, :)) +
c2 .* r2 .* (gbest_x - x(i, :));
        x(i, :) = x(i, :) + v(i, :);

        % Enforce search space bounds
        x(i, :) = min(x(i, :), ub);
        x(i, :) = max(x(i, :), lb);
    end

    % Update personal best positions and costs
    for i = 1:num_particles
        cost = objective(x(i, :));
        if abs(cost - target_value) < abs(pbest_cost - target_value)%cost
< pbest_cost(i)
            pbest_x(i, :) = x(i, :);
            pbest_cost(i) = cost;
        end
    end

    % Update global best position and cost
    [min_cost, min_index] = min(pbest_cost);
    if abs(min_cost - target_value) < abs(gbest_cost - target_value)
        gbest_cost = min_cost;
        ideal_reached=false;
        if abs(gbest_cost - target_value) <= 0.1 && ideal_reached==false
            IdealIter=iter;
            ideal_reached=true;
        end
        gbest_x = pbest_x(min_index, :);
    end

    % Print current iteration and best cost
    fprintf('Iteration %d: Best cost = %.4f\n', iter, gbest_cost);
    %onetime = true;
    %if gbest_cost == target_value && onetime
        %fprintf('\nIdeal Power Reached in second: %d\n',iter);
        %onetime = false;
    %end
    disp(gbest_x);
    power(iter) = gbest_cost;
    %pause(1);
end

% Print final results
fprintf('\nFinal results:\n');
fprintf('Best parameters =
[%.2f %.2f %.2f %.2f %.2f %.2f %.2f %.2f %.2f %.2f]\n', gbest_x);
fprintf('Best cost = %.4f\n', gbest_cost);
timesaved = iter-IdealIter;

% Plot the desired and actual power output over time using the best
parameters
t_desired = [0 10 20 30 40 50 60 70 80 90 100];

```

```

P_desired = [0 0.2 0.3 0.4 0.6 0.8 1.0 0.9 0.8 0.6 0.4];
t_actual = linspace(0, 100, 11);
P_actual = zeros(size(t_actual));
P_actual(t_actual >= gbest_x(1) & t_actual < gbest_x(2)) =
(t_actual(t_actual >= gbest_x(1) & t_actual < gbest_x(2)) - gbest_x(1)) /
(gbest_x(2) - gbest_x(1));
P_actual(t_actual >= gbest_x(2) & t_actual < gbest_x(3)) = 1;
P_actual(t_actual >= gbest_x(3) & t_actual < gbest_x(4)) = 1 -
(t_actual(t_actual >= gbest_x(3) & t_actual < gbest_x(4)) - gbest_x(3)) /
(gbest_x(4) - gbest_x(3));
P_actual(t_actual >= gbest_x(4) & t_actual < gbest_x(5)) = 0.5 * (1 +
cos(pi * (t_actual(t_actual >= gbest_x(4) & t_actual < gbest_x(5))) /
(gbest_x(5) - gbest_x(4))));

% Find the position of each particle where the best cost is achieved
best_positions = x;

% Display the positions
disp('Particle positions where best cost is achieved:');
disp(best_positions);

%Define the objective function
function cost = objective(x)
    %Define the desired power output over time
    x1 = x(:,1);
    x2 = x(:,2);
    x3 = x(:,3);
    x4 = x(:,4);
    x5 = x(:,5);
    x6 = x(:,6);
    x7 = x(:,7);
    x8 = x(:,8);
    x9 = x(:,9);
    x10 = x(:,10);
    x11 = x(:,11);
    x12 = x(:,12);
    x13 = x(:,13);
    x14 = x(:,14);
    x15 = x(:,15);
    x16 = x(:,16);
    x17 = x(:,17);
    x18 = x(:,18);
    x19 = x(:,19);

total_process_duration=@(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x1
5,x16,x17,x18,x19) -26.20 * x1 + 0.00 * x2 + -0.39 * x3 + -5.69 * x4 + -
0.16 * x5 + 0.29 * x6 + -0.03 * x7 + 0.18 * x8 + -6.42 * x9 + 1.65 * x10 +
5.06 * x11 + 4.50 * x12 + 1.10 * x13 + -0.06 * x14 + -0.78 * x15 + -0.37 *
x16 + 3.37 * x17 + -0.29 * x18 + 0.19 * x19 + -21507.32;%-21505.42;
    total_process_duration2=@(P_fuel2,T_steam2,P_inlet2,P_outlet2,Speed2)
(((P_outlet2*P_inlet2)/T_steam2)/(2*P_fuel2))*Speed2+60;

    wt = 0.06;
    wt2 = 0.3;

```

```

duration_total=total_process_duration(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x
12,x13,x14,x15,x16,x17,x18,x19);

    % Calculate the weighted sum of process durations
    cost = duration_total;%+wt2*duration_total2;
end

```

## Appendix B: Data Analysis Sample Coding for Firing Sequence of Cold Startup

```

sheet = 1;
xlRangex = 'C3:U31';
xlRangey = 'B3:B31';
xlRangetime = 'B2:B227';
independent = xlsread('Firing Cold.xlsx',sheet, xlRangex);
dependent = xlsread('Firing Cold.xlsx',sheet, xlRangey);
timeline = 1:size(dependent);
%xlsread('Initiate Cold.xlsx',sheet, xlRangetime);

% Define the degree of the polynomial you want to fit
degree = 1; % You can change this to the desired degree

% Fit the polynomial
coefficients = polyfitn(independent, dependent, degree);

% Evaluate the polynomial fit
yFit = polyvaln(coefficients, independent);

coef = coefficients.Coefficients;
varnum = size(coef,2);
coefficientPairs = cell(1, varnum);
for i = 1:varnum
    coefficientPairs{i} = sprintf('%.2f * x%d', coef(i), i);
end
coefficientString = strjoin(coefficientPairs, ' + ');
fprintf('y = %s\n', coefficientString);
curve=power;

```

## Appendix C: Display GUI Sample Coding

```

classdef App < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure          matlab.ui.Figure
        MeanErrorEditField matlab.ui.control.NumericEditField
        MeanErrorEditFieldLabel matlab.ui.control.Label
        RSSEditField       matlab.ui.control.NumericEditField
        RSSEditFieldLabel  matlab.ui.control.Label
        SequenceButtonGroup matlab.ui.container.ButtonGroup
        IPCOnButton         matlab.ui.control.ToggleButton
        WarmUpButton        matlab.ui.control.ToggleButton
        TempMatchButton     matlab.ui.control.ToggleButton
    end
end

```

```

FSNLButton          matlab.ui.control.ToggleButton
SyncButton          matlab.ui.control.ToggleButton
FiringButton        matlab.ui.control.ToggleButton
PurgeButton         matlab.ui.control.ToggleButton
InitiateButton      matlab.ui.control.ToggleButton
TimeSavesdEditField matlab.ui.control.NumericEditField
TimeSavesdEditFieldLabel matlab.ui.control.Label
StartupTypeButtonGroup matlab.ui.container.ButtonGroup
HotButton           matlab.ui.control.ToggleButton
Warm2Button         matlab.ui.control.ToggleButton
Warm1Button         matlab.ui.control.ToggleButton
ColdButton         matlab.ui.control.ToggleButton
FormulaTextArea     matlab.ui.control.TextArea
FormulaTextAreaLabel matlab.ui.control.Label
UITable            matlab.ui.control.Table
UIAxes             matlab.ui.control.UIAxes
end

% Callbacks that handle component events
methods (Access = private)

    % Callback function
    function ButtonValueChanged(app, event)
        value = app.Button.Value;
        %[filename,path]=uigetfile();
        temp=readtable("Initiate Cold.xlsx");
        app.UITable.Data = temp;
        figure(app.UIFigure);
    end

    % Display data changed function: UITable
    function UITableDisplayDataChanged(app, event)
        newDisplayData = app.UITable.DisplayData;
    end

    % Value changed function: FormulaTextArea
    function FormulaTextAreaValueChanged(app, event)
        value = app.FormulaTextArea.Value;
    end

    % Button down function: UIAxes
    function UIAxesButtonDown(app, event)

    end

    % Selection changed function: StartupTypeButtonGroup
    function StartupTypeButtonGroupSelectionChanged(app, event)
        selectedButton = app.StartupTypeButtonGroup.SelectedObject;
        if app.ColdButton.Value==1
            temp=readtable("Labelled
Data.xlsx",ReadRowNames=true,Sheet="Cold Start Sample");
            app.UITable.Data = temp;
        elseif app.Warm1Button.Value==1
            temp=readtable("Labelled
Data.xlsx",ReadRowNames=true,Sheet="Warm I Start Sample");
            app.UITable.Data = temp;
        elseif app.Warm2Button.Value==1
            temp=readtable("Labelled
Data.xlsx",ReadRowNames=true,Sheet="Warm I IStart Sample");

```



```

        app.UITable.Data = temp;
    elseif app.HotButton.Value==1
        temp=readtable("Labelled
Data.xlsx",ReadRowNames=true,Sheet="Hot Start Sample");
        app.UITable.Data = temp;
    end
end

% Value changed function: TimeSavedsEditField
function TimeSavedsEditFieldValueChanged(app, event)
    value = app.TimeSavedsEditField.Value;
end

% Selection changed function: SequenceButtonGroup
function SequenceButtonGroupSelectionChanged(app, event)
    selectedButton = app.SequenceButtonGroup.SelectedObject;
    if app.PurgeButton.Value==1 && app.ColdButton.Value==1
        Purge_Cold_PSO;
        Purge_Cold;
        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y,'--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');
        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
    elseif app.InitiateButton.Value==1 && app.ColdButton.Value==1
        Approx_cost;
        Initiate_Cold;
        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y,'--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');
        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
    elseif app.FiringButton.Value==1 && app.ColdButton.Value==1
        Firing_Cold_PSO;
        Firing_Cold;
        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y,'--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');

```

```

elseif app.WarmUpButton.Value==1 && app.ColdButton.Value==1
    WarmUp_Cold_PSO;
    WarmUp_Cold;
    cla(app.UIAxes);
    formula = coefficientString;
    app.FormulaTextArea.Value = formula;
    app.TimeSavedsEditField.Value=timesaved;
    y=curve;
    plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
    title(app.UIAxes,"Power over Time");
    legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');
elseif app.SyncButton.Value==1 && app.ColdButton.Value==1
    cla(app.UIAxes);
    app.FormulaTextArea.Value = "No Data";
elseif app.FSNLButton.Value==1 && app.ColdButton.Value==1
    cla(app.UIAxes);
    app.FormulaTextArea.Value = "No Data";
elseif app.TempMatchButton.Value==1 && app.ColdButton.Value==1
    cla(app.UIAxes);
    app.FormulaTextArea.Value = "No Data";
elseif app.IPCOnButton.Value==1 && app.ColdButton.Value==1
    cla(app.UIAxes);
    app.FormulaTextArea.Value = "No Data";
elseif app.InitiateButton.Value==1 && app.Warm1Button.Value==1
    Initiate_Warm_1_PSO;
    Initiate_Warm_1;
    cla(app.UIAxes);
    formula = coefficientString;
    app.FormulaTextArea.Value = formula;
    app.TimeSavedsEditField.Value=timesaved;
    y=curve;
    plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
    title(app.UIAxes,"Power over Time");
    legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');
    app.RSSEditField.Value = RSS;
    app.MeanErrorEditField.Value = MeanError;
elseif app.PurgeButton.Value==1 && app.Warm1Button.Value==1
    Purge_Warm1_PSO;
    Purge_Warm1;
    cla(app.UIAxes);
    formula = coefficientString;
    app.FormulaTextArea.Value = formula;
    app.TimeSavedsEditField.Value=timesaved;
    y=curve;
    plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
    title(app.UIAxes,"Power over Time");
    legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');
    app.RSSEditField.Value = RSS;
    app.MeanErrorEditField.Value = MeanError;
elseif app.SyncButton.Value==1 && app.Warm1Button.Value==1
    Sync_Warm_PSO;
    Sync_Warm;
    cla(app.UIAxes);

```

```

        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');

        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
elseif app.InitiateButton.Value==1 && app.Warm2Button.Value==1
        Initiate_Warm_2_PSO;
        Initiate_Warm_2;
        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');

        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
elseif app.PurgeButton.Value==1 && app.Warm2Button.Value==1
        Purge_Warm2_PSO;
        Purge_Warm2;
        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');

        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
elseif app.InitiateButton.Value==1 && app.HotButton.Value==1
        Initiate_Hot_PSO;
        Initiate_Hot;
        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavedsEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y, '--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');

        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
elseif app.PurgeButton.Value==1 && app.HotButton.Value==1
        Purge_Hot_PSO;
        Purge_Hot;

```

```

        cla(app.UIAxes);
        formula = coefficientString;
        app.FormulaTextArea.Value = formula;
        app.TimeSavesEditField.Value=timesaved;
        y=curve;
        plot(app.UIAxes,timeline, y,'--
or',timeline,dependent,'g',timeline,yFit,'b');
        title(app.UIAxes,"Power over Time");
        legend(app.UIAxes,'Optimized Data','Original Data','Fitted
Curve');

        app.RSSEditField.Value = RSS;
        app.MeanErrorEditField.Value = MeanError;
    end
end

% Value changed function: RSSEditField
function RSSEditFieldValueChanged(app, event)
    value = app.RSSEditField.Value;
end

% Value changed function: MeanErrorEditField
function MeanErrorEditFieldValueChanged(app, event)
    value = app.MeanErrorEditField.Value;
end
end

% Component initialization
methods (Access = private)

% Create UIFigure and components
function createComponents(app)

% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible','off');
app.UIFigure.Position = [100 100 640 480];
app.UIFigure.Name = 'MATLAB App';

% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
xlabel(app.UIAxes, 'Timeline')
ylabel(app.UIAxes, 'Power')
zlabel(app.UIAxes, 'Z')
app.UIAxes.ButtonDownFcn = createCallbackFcn(app,
@UIAxesButtonDown, true);
app.UIAxes.Position = [33 64 300 185];

% Create UITable
app.UITable = uitable(app.UIFigure);
app.UITable.ColumnName = {'Column 1'; 'Column 2'; 'Column 3';
'Column 4'; 'Column 5'; 'Column 1'; 'Column 2'; 'Column 3'; 'Column 4';
'Column 5'; 'Column 1'; 'Column 2'; 'Column 3'; 'Column 4'; 'Column 5';
'Column 16'};
app.UITable.RowName = {};
app.UITable.DisplayDataChangedFcn = createCallbackFcn(app,
@UITableDisplayDataChanged, true);
app.UITable.Position = [320 273 302 185];

% Create FormulaTextAreaLabel
app.FormulaTextAreaLabel = uilabel(app.UIFigure);

```

```

app.FormulaTextAreaLabel.HorizontalAlignment = 'right';
app.FormulaTextAreaLabel.Position = [44 434 49 22];
app.FormulaTextAreaLabel.Text = 'Formula';

% Create FormulaTextArea
app.FormulaTextArea = uitextarea(app.UIFigure);
app.FormulaTextArea.ValueChangedFcn = createCallbackFcn(app,
@FormulaTextAreaValueChanged, true);
app.FormulaTextArea.Position = [108 398 150 60];

% Create StartupTypeButtonGroup
app.StartupTypeButtonGroup = uibuttongroup(app.UIFigure);
app.StartupTypeButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @StartupTypeButtonGroupSelectionChanged, true);
app.StartupTypeButtonGroup.Title = 'Start-up Type';
app.StartupTypeButtonGroup.Position = [422 140 123 123];

% Create ColdButton
app.ColdButton = uitogglebutton(app.StartupTypeButtonGroup);
app.ColdButton.Text = 'Cold';
app.ColdButton.Position = [11 69 100 23];
app.ColdButton.Value = true;

% Create Warm1Button
app.Warm1Button = uitogglebutton(app.StartupTypeButtonGroup);
app.Warm1Button.Text = 'Warm 1';
app.Warm1Button.Position = [11 48 100 23];

% Create Warm2Button
app.Warm2Button = uitogglebutton(app.StartupTypeButtonGroup);
app.Warm2Button.Text = 'Warm 2';
app.Warm2Button.Position = [12 27 100 23];

% Create HotButton
app.HotButton = uitogglebutton(app.StartupTypeButtonGroup);
app.HotButton.Text = 'Hot';
app.HotButton.Position = [11 5 100 23];

% Create TimeSavedsEditFieldLabel
app.TimeSavedsEditFieldLabel = uilabel(app.UIFigure);
app.TimeSavedsEditFieldLabel.HorizontalAlignment = 'right';
app.TimeSavedsEditFieldLabel.Position = [48 24 86 22];
app.TimeSavedsEditFieldLabel.Text = 'Time Saved (s)';

% Create TimeSavedsEditField
app.TimeSavedsEditField = uieditfield(app.UIFigure,
'numeric');
app.TimeSavedsEditField.ValueChangedFcn =
createCallbackFcn(app, @TimeSavedsEditFieldValueChanged, true);
app.TimeSavedsEditField.Position = [149 24 100 22];

% Create SequenceButtonGroup
app.SequenceButtonGroup = uibuttongroup(app.UIFigure);
app.SequenceButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @SequenceButtonGroupSelectionChanged, true);
app.SequenceButtonGroup.Title = 'Sequence';
app.SequenceButtonGroup.Position = [12 262 291 117];

% Create InitiateButton

```

```

app.InitiateButton = uitogglebutton(app.SequenceButtonGroup);
app.InitiateButton.Text = 'Initiate';
app.InitiateButton.Position = [38 69 100 23];
app.InitiateButton.Value = true;

% Create PurgeButton
app.PurgeButton = uitogglebutton(app.SequenceButtonGroup);
app.PurgeButton.Text = 'Purge';
app.PurgeButton.Position = [38 48 100 23];

% Create FiringButton
app.FiringButton = uitogglebutton(app.SequenceButtonGroup);
app.FiringButton.Text = 'Firing';
app.FiringButton.Position = [38 27 100 23];

% Create SyncButton
app.SyncButton = uitogglebutton(app.SequenceButtonGroup);
app.SyncButton.Text = 'Sync';
app.SyncButton.Position = [141 69 100 23];

% Create FSNLButton
app.FSNLButton = uitogglebutton(app.SequenceButtonGroup);
app.FSNLButton.Text = 'FSNL';
app.FSNLButton.Position = [141 48 100 23];

% Create TempMatchButton
app.TempMatchButton = uitogglebutton(app.SequenceButtonGroup);
app.TempMatchButton.Text = 'TempMatch';
app.TempMatchButton.Position = [141 27 100 23];

% Create WarmUpButton
app.WarmUpButton = uitogglebutton(app.SequenceButtonGroup);
app.WarmUpButton.Text = 'WarmUp';
app.WarmUpButton.Position = [39 6 100 23];

% Create IPCOnButton
app.IPCOnButton = uitogglebutton(app.SequenceButtonGroup);
app.IPCOnButton.Text = 'IPCOn';
app.IPCOnButton.Position = [141 6 100 23];

% Create RSSEditFieldLabel
app.RSSEditFieldLabel = uilabel(app.UIFigure);
app.RSSEditFieldLabel.HorizontalAlignment = 'right';
app.RSSEditFieldLabel.Position = [411 101 30 22];
app.RSSEditFieldLabel.Text = 'RSS';

% Create RSSEditField
app.RSSEditField = uieditfield(app.UIFigure, 'numeric');
app.RSSEditField.ValueChangedFcn = createCallbackFcn(app,
@RSSEditFieldValueChanged, true);
app.RSSEditField.Position = [456 101 100 22];

% Create MeanErrorEditFieldLabel
app.MeanErrorEditFieldLabel = uilabel(app.UIFigure);
app.MeanErrorEditFieldLabel.HorizontalAlignment = 'right';
app.MeanErrorEditFieldLabel.Position = [379 55 65 22];
app.MeanErrorEditFieldLabel.Text = 'Mean Error';

% Create MeanErrorEditField

```

```

    app.MeanErrorEditField = uicontrol(app.UIFigure, 'numeric');
    app.MeanErrorEditField.ValueChangedFcn =
createCallbackFcn(app, @MeanErrorEditFieldValueChanged, true);
    app.MeanErrorEditField.Position = [459 55 100 22];

    % Show the figure after all components are created
    app.UIFigure.Visible = 'on';
end
end

% App creation and deletion
methods (Access = public)

    % Construct app
    function app = App

        % Create UIFigure and components
        createComponents(app)

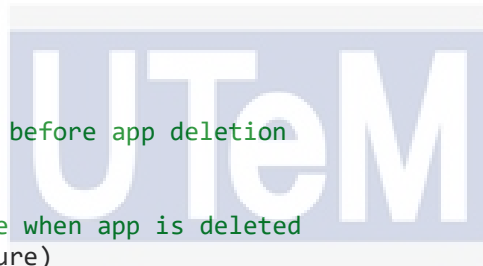
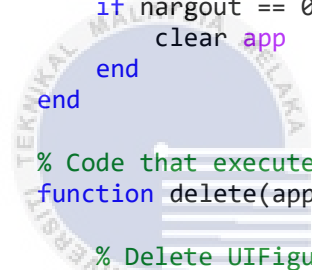
        % Register the app with App Designer
        registerApp(app, app.UIFigure)

        if nargin == 0
            clear app
        end
    end

    % Code that executes before app deletion
    function delete(app)

        % Delete UIFigure when app is deleted
        delete(app.UIFigure)
    end
end
end

```



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA